



Carl von Ossietzky Universität Oldenburg
Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Map-based Localization for Automated Vehicles using LiDAR Features

Von der Fakultät für Informatik, Wirtschafts- und
Rechtswissenschaften der Carl von Ossietzky Universität Oldenburg
zur Erlangung des Grades und Titels einer

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

angenommene Dissertation

von Frau Constanze Hungar

geboren am 07.05.1993 in Oldenburg

Gutachter	Prof. Dr. Frank Köster
Weitere Gutachter	Prof. Dr. Stephan Schmidt
Tag der Disputation	09.07.2021

Kurzfassung

Lokalisierung spielt eine wichtige Rolle für die Entwicklung automatisierter Fahrzeuge. Das ist der Fall, da sich Teilprobleme des automatisierten Fahrens, wie die Prädiktion von dynamischen Objekten und die Trajektorienplanung, auf Karteninformationen stützen. Dadurch ist eine Karten-relative Lokalisierung unabdingbar. Zusätzlich wird in vielen Fällen eine globale Ortung der Fahrzeuge im Zusammenhang mit kooperativen Fahrfunktionen, wie z. B. Fahrzeug-zu-Fahrzeug-Kommunikation, benötigt.

Eine Lokalisierung, die eine globale und eine Karten-relative Fahrzeugpose berechnet, kann auf verschiedene Weisen gelöst werden. Häufig ruhen Lokalisierungslösungen auf statischen, semantischen Objekten, wie Straßenschildern oder Häuserkanten, sog. Landmarken. Um unabhängig von diesen Infrastrukturelementen zu sein, wird in dieser Arbeit ein generalisiertes Konzept zur Realisierung einer Feature-basierten Lokalisierung anhand von Light Detection and Ranging (LiDAR) Sensoren, genannt *LiDAR-Feature-basierte Lokalisierung*, entwickelt. Das Konzept dieser Arbeit kommt ohne präzise Semantik der Feature aus.

Hierfür werden nicht-semantische Muster für jeden Punkt der LiDAR-Punktwolke bestimmt. Ein solches Muster eines Punktes ist definiert durch eine Deskription, welcher charakteristische Beziehungen zwischen Abtastpunkten innerhalb der lokalen Nachbarschaft um den jeweiligen Punkt beschreibt. Diese Deskription wird durch einen Algorithmus, welcher Deskriptor genannt wird, ermittelt. Da sich nicht jeder Punkt einer Punktwolke für eine robuste Lokalisierung eignet, wird in dieser Arbeit ein automatisiertes Verfahren zur Detektion signifikanter und persistenter Charakteristiken, genannt Feature, vorgeschlagen und umgesetzt. Dabei verwendet das Verfahren die Deskriptionen und Positionen der Punkte, um die Eignung der Punkte zu überprüfen. Das Extrahieren der Feature findet sowohl zu Laufzeit im Fahrzeug als auch offline zur Erstellung einer global referenzierten Feature-Karte statt. Ein Abgleich zwischen den im Fahrzeug detektierten und aus der Karte gelesenen Feature liefert die fahrzeugeigene Pose. Um die Praktikabilität des Feature-basierten Lokalisierungsansatzes zu testen, werden die Feature in einem dem Stand der Technik entsprechenden Lokalisierungsalgorithmus integriert. Dafür wird in dieser Arbeit ein Graphen-basierter Ansatz gewählt.

Mit gründlichen Analysen und einem Fokus auf die Umsetzung in Realdaten wird untersucht, inwiefern die Verwendung von nicht-semantischen Feature einen Nutzen für eine funktionsfähige Lokalisierung hat. Dafür wurden eigene entwickelt und bereits existierende Deskriptoren, die häufig lediglich anhand von synthetischen Daten, wie dicht abgetasteten Hasenmodelle, untersucht wurden, auf deren Tauglichkeit für den Einsatz in Verkehrsszenarien analysiert. Es kann gezeigt werden, dass Deskriptoren aus dem Stand der Technik einige der praxisrelevanten Kriterien erfüllen. Für die Anforderungen, welche die Deskriptoren nicht einhalten, werden Erweiterungen, eine Handhabung mit diesen oder neue Entwicklungen vorgestellt. Für die eigens entwickelten Deskriptoren, genannt *DeLL*, *GRAIL* und *modified HoPD*, wird empirisch demonstriert, dass sie einige Eigenschaften der existierenden Deskriptoren im Praxiseinsatz verbessern, wie zum Beispiel die Distanzunabhängigkeit. Des Weiteren wird bestätigt, dass das entwickelte Feature Detektionsverfahren für eine robuste Lokalisierung geeignete Elemente extrahiert. Hierbei wird ein bestehendes auf Deskriptionen basierendes Extraktionsverfahren, welches nicht eigens für eine Lokalisierung entworfen wurde, zum empirischen Vergleich herangezogen. Es wird anhand von Realdaten bestätigt, dass die in dieser Arbeit realisierte Methode deutlich besser für die Lokalisierung geeignet ist. Ebenso wird mit realen Punktwolken unterschiedlicher Szenarien demonstriert, dass Szenerie-unabhängig Feature detektiert werden können.

Die Untersuchungen der *LiDAR-Feature-basierte Lokalisierung* zeigen, dass der gewählte nicht-semantische Ansatz für die Praxis tauglich ist. Das bedeutet, dass eine Posengenauigkeit erreicht wird, welche die einer Lösung basierend auf Satelliteninformationen (Globale Navigations Satellitensysteme) unterbietet. Dies steht dem hohen Rechenaufwand einer nicht-semantischen Lokalisierung gegenüber. Ein Vorteil des Algorithmus, der experimentell gestützt wird, stellt die Einsetzbarkeit der Lokalisierung in beliebigen, nicht vordefinierten Gebieten dar.

Abstract

Localization is crucial for the development of automated vehicles. This is the case because subproblems of automated driving, such as the prediction of dynamic objects and trajectory planning, rely on prior knowledge in form of map information. This means that a map-relative localization is required. In addition, in many cases, a global positioning of vehicles considering cooperative driving functions, such as vehicle-to-vehicle communication, is needed.

Localization which calculates a global and a map-relative vehicle pose can be solved in various ways. Localization solutions often rely on static, semantic objects, like road signs or house corners, called landmarks. In order to be independent of these infrastructure elements a generalized concept for the realization of non-semantic localization relying on data sampled with Light Detection and Ranging (LiDAR) sensors, called *LiDAR-Feature-based Localization*, is developed in this thesis. The concept of this thesis manages without precise semantics of the feature.

For this purpose, non-semantic patterns are determined for each point of the LiDAR point cloud. Such a pattern of a point is defined by a description which depicts characteristic relations between sampling points within the local neighborhood around the respective point. This description is determined by an algorithm called descriptor. Since not every point of a point cloud is suitable for a robust localization, an automated method for the detection of significant and persistent characteristics, called feature, is proposed and implemented in this thesis. The procedure uses the descriptions and positions of the points to check the suitability of the points. The feature is extracted both during runtime on-board the vehicle and offline to create a globally referenced feature map. A comparison between the features detected on-board and those read from the map provides the vehicle's pose. In order to test the practicability of the feature-based localization approach, the features are integrated in a localization algorithm corresponding to the current state of the art. For this purpose, a graph-based Simultaneous Localization And Mapping (SLAM) approach is chosen in this work.

With thorough analyzes and a focus on the realization in real data, it is examined to what extent the use of non-semantic features has a benefit for a functional localization. For this, own descriptors were developed and already existing descriptors, which were often only examined on the basis of synthetic data, such as densely scanned rabbit models, were analyzed for their suitability for use in traffic scenarios. It can be shown that state-of-the-art descriptors already meet many of the criteria relevant in practice. For the requirements that the descriptors do not comply with, expansions, handling of these or new developments are presented. For the specially developed descriptors, called *DeLL*, *GRAIL*, and *modified HoPD*, it is empirically demonstrated that they improve some properties of the existing descriptors in practice, such as distance independence. It is also confirmed that the feature detection method extracts suitable elements for robust localization. An existing extraction process based on descriptions, which was not specially designed for localization, is used for the empirical comparison. It is confirmed on the basis of real data that the method implemented in this work is significantly better suited for localization. Also, real point clouds of different sceneries are used to demonstrate that features can be detected independently of the scenery.

The investigations of the *LiDAR-Feature-based Localization* show that the selected non-semantic approach is suitable for practice. This means that a positional accuracy is achieved which outperforms that of a solution based on satellite information (Global Navigation Satellite Systems). However, this contrasts with the high computational effort of non-semantic localization. An advantage of the algorithm, which is experimentally supported, is that localization can be used in any desired, non-predefined domains.

Acknowledgment

The results of my work as a PhD-student at the department *Localization and Map* of the Volkswagen Group Research in cooperation with the department *Intelligente Transportsysteme* of the Carl von Ossietzky Universität Oldenburg are presented in this thesis.

First of all, I would like to take this opportunity to thank Prof. Dr. Frank Köster for supervising me during my PhD. His regular feedback and creative discussions have greatly improved my work. With his structured comments, he also helped drive the constant progress of my research.

I would also like to thank my sub-department leader at the Volkswagen AG Dr. Bernd Rech, who always supported me and this thesis' topic while keeping the tasks that would have hindered me in my work away from me. His advice and support have helped me a lot throughout the entire time. A big thank you also goes to Dr. Stefan Jürgens. He was a great help, especially in challenges regarding this thesis, as were his reviews of my publications and this work. I would like to offer my special thanks to my colleagues and students of the department for their contributions and effort.

Throughout the whole time as a PhD student, the team members working at the localization and map project were a reliable help for the implementation of the designed concept to whom I would like to extend my sincere thanks to. Especially, I would like to thank the team members at my sub-department for sharing their expertise. My special thanks go to Daniel Laubrich and Jonas Jung without their implementation expertise and vehicle integration tips I would not have been able to advance this quickly. They were always ready to answer my questions until I had none.

I am equally grateful to my colleagues in the department *Automated Driving* at Volkswagen Group Research. They have created a friendly and open-minded work environment in which I felt very welcome and comfortable.

Lastly, my gratitude needs to be expressed to my family and friends for their unwavering support and belief in me. I could always rely on them and they were there to listen to my doubts and complaints and to encourage me. Especially, I want to thank my father Priv. Doz. Dr. Hardi Hungar, who gave me the best tips on how to work scientifically and always supported me.

Disclaimer

The results, opinions and conclusions expressed in this thesis are not necessarily those of Volkswagen Aktiengesellschaft.

Ergebnisse, Meinungen und Schlüsse dieser Dissertation sind nicht notwendigerweise die der Volkswagen Aktiengesellschaft.

Statement

I completed the work independently and used only the indicated resources.
Ich habe die Arbeit selbständig verfasst und nur die angegebenen Hilfsmittel benutzt.

List of Publications

- [54] C. Hungar, F. Köster, and S. Jürgens. Ein Beitrag zur Karten-basierten Positionierung von Fahrzeugen mittels Mustererkennung in LiDAR-Daten. *AAET Automatisiertes und vernetztes Fahren*, pages 135–155, 2019
- [52] C. Hungar, S. Brakemeier, S. Jürgens, and F. Köster. GRAIL: A gradients-of-intensities-based local descriptor for map-based localization using LiDAR sensors. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 4398–4403, 2019
- [53] C. Hungar, J. Fricke, S. Jürgens, and F. Köster. Detection of feature areas for map-based localization using LiDAR descriptors. In *IEEE Workshop on Positioning, Navigation and Communications (WPNC)*, pages 1–6, 2019
- [55] C. Hungar, S. Jürgens, D. Wilbers, and F. Köster. Map-based localization with factor graphs for automated driving using non-semantic lidar features. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020

Contents

Acronyms and Symbols	XIX
I. Context of LiDAR-Feature-based Localization	1
1. Introduction	3
1.1. Motivating LiDAR-Feature-based Localization	3
1.2. Introducing Feature-based Localization	5
1.3. Contributions	7
1.4. Overview	8
2. Related Work and Basics	11
2.1. Relevant Literature	11
2.1.1. LiDAR Descriptors	11
2.1.2. Extraction of Non-Semantic Elements	19
2.1.3. Localization with Non-Semantic LiDAR Data	27
2.1.4. Conclusions from Relevant Literature for this Thesis	32
2.2. Mathematical Fundamentals	35
2.2.1. Coordinate Systems	35
2.2.2. Principal Component Analysis	36
2.2.3. Iterative Closest Point	37
2.2.4. Ramer-Douglas-Peucker Algorithm	39
2.2.5. Daisy Descriptor	39
2.2.6. Density-based Spatial Clustering of Applications with Noise	42
2.2.7. k-Medoids	43
2.2.8. Graph-based Simultaneous Localization and Mapping	44
II. Describing and Selecting Features	51
3. LiDAR-Based Descriptors	53
3.1. Terms and Methodology	53
3.1.1. Defining the Description Computation	53
3.1.2. Procedure of Descriptor Evaluations and Extensions	54
3.1.3. Research Questions of the Application of Descriptors	59
3.2. Geometry-Based Descriptors	60
3.2.1. Analyzes of Geometry-Based Descriptors in Real-World Environment	60
3.2.2. Own Developments and Extensions	74
3.3. Intensity-Based Descriptors	88
3.3.1. Theoretical Analyzes on State-of-the-Art Algorithms	88
3.3.2. Developments	89
3.3.3. Analysis	91

3.4. Summary	92
4. Feature Area Extractions	95
4.1. Terms and Methodology	95
4.1.1. Defining the Feature Extraction	95
4.1.2. Procedure of Feature Area Extraction	97
4.1.3. Research Questions of the Application of Feature Extraction	101
4.2. Generation of Feature Areas in Non-Semantic Way	102
4.2.1. Preprocessing	102
4.2.2. Description Clustering	103
4.2.3. Spatial Clustering	105
4.3. Extraction of Map and On-Board FAs applying the LIG	105
4.4. Implementation of Extraction Process	110
4.4.1. Implementation of Description Clustering	111
4.4.2. Implementation of Euclidean Clustering	114
4.4.3. Implementation LIG’s Distinctiveness	115
4.4.4. Map Data Format	117
4.5. Analyzes of Extraction Process Using Real-World Data	117
4.5.1. Position Accuracy Matching Map and On-Board Feature Areas	118
4.5.2. Persistence of Extracted Map and On-Board Feature Areas	119
4.5.3. Scenery Coverage	120
4.5.4. Compressibility of Map Feature Area Extraction	122
4.6. Summary	123
III. Positioning with Features	125
5. Localization using LiDAR Feature Areas	127
5.1. Terms and Methodology	127
5.1.1. Defining the Localization	127
5.1.2. Procedure of Localization with LiDAR Feature Areas	128
5.1.3. Research Questions of the Application of the Localization Algorithm	131
5.2. Integration of Features into Graph-Based SLAM	131
5.2.1. Collection of On-Board Feature Areas and Map Generation	132
5.2.2. Association of Feature Areas	133
5.2.3. Map Matching	135
5.2.4. Graph Building	137
5.2.5. Graph Optimization	138
5.2.6. Sliding Window	142
5.3. Analyzes	143
5.3.1. Position Accuracy	143
5.3.2. Demand for Computing Resources	145
5.4. Summary	152
6. Conclusion	153
6.1. Key Findings and Contributions	153
6.2. Future Work	155
List of Figures	169

List of Tables	172
Appendices	175
A. AI-Based Descriptors	177
B. Number of Clusters in Description Space	187
C. Feature Area Extraction with Intensity-based Descriptors	189

Acronyms and Symbols

Acronyms

ABS	Anti-Lock Braking System
AI	Artificial Intelligence
B-SHOT	Binary Signatures of Histograms of Orientations
CAD	Computer-Aided Design
CAE	Convolutional Auto-Encoder
CAE-LO	Convolutional Auto-Encoder based LiDAR Odometry
CLARA	Clustering LARge Applications
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DELIGHT	DESCRIPTOR of LiDAR Intensities as a Group of Histograms
DeLL	Depth Leap Local
DoF	Degree of Freedom
DOG	Difference Of Gaussians
DON	Difference Of Normal
EKF	Extended Kalman Filter
ENU	East North Up
ESP	Electronic Stability Program
FA	Feature Area
FAR	Feature Area Representative
FPFH	Fast Point Feature Histogram
FWHM	Full Width at Half Maximum
GLARE	Geometric Landmark Relations
GLONASS	GLOBAL Navigation Satellite System
GNSS	Global Navigation and Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit

Contents

GRAIL	GRAdients of Intensities as a Local descriptor
HOG	Histogram of Oriented Gradients
HoPD	Histogram of Point Distributions
HT	Hough Transformation
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
ISHOT	Intensity Signatures of Histograms of Orientations
ISS	Intrinsic Shape Signatures
KPQ	Key Point Quality
LiDAR	Light Detection And Ranging
LIG	Localization Information Gain
LRF	Local Reference Frame
LSP	Local Surface Patches
LTP	Local Tangential Plane
MAP	Maximum A Posteriori
MeshDOG	Mesh Difference Of Gaussians
MeshHOG	Mesh Histogram of Oriented Gradients
MFA	Map Feature Area
MLP	Multi-Layer Perception
MSER	Maximally Stable Extremal Regions
NARF	Normal Aligned Radial Feature
OFA	On-board Feature Area
PC	Personal Computer
PCA	Principal Component Analysis
PCL	Point Cloud Library
PFH	Point Feature Histogram
RADAR	Radio Detection And Ranging
RAM	Random Access Memory
RANSAC	RANdom SAmples Consensus
RMS	Root Mean Square
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RoPS	Rotational Projection of Statistics

RTK	Real-Time Kinematic
SDS	Self-Driving System
SHOT	Signatures of Histograms of Orientations
SI	Spin Images
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SOD	Sum Of Differences
SOM	Self-Organizing Map
SPFH	Simplified Point Feature Histogram
SRF	Sensor Reference Frame
STN	Spatial Transformation Network
SURF	Speeded Up Robust Feature
T-Net	Transformation Network
TriSI	Tri-Spin-Image
USC	Unique Shape Context
UTM	Universal Transverse Mercator
V2X	vehicle-to-everything
VD-LSD	Variable Dimensional Local Shape Descriptor
VRF	Vehicle Reference Frame
WGS84	World Geodetic System of 1984
3DHoPD	3D Histogram of Point Distribution
3DSC	3D Shape Context

Symbols

Matrices are denoted in upper case bold letters (\mathbf{A}), vectors are represented as lower case bold letters (\mathbf{a}), corresponding elements are denoted in lower case using the same letter (a), sets are marked with an upper case calligraphic letter (\mathcal{A}). Transpose operations are marked with a superscripted $(\cdot)^T$. The power set of a set is denoted with $\mathcal{P}(\cdot)$.

Symbol	Unit	Description
\mathcal{P}	–	Point cloud with geometry and intensity information
$\mathcal{P}^{x,y,z}$	–	Point cloud with geometry information
\mathcal{P}^i	–	Point cloud with intensity information
\mathbf{p}	$m \times m \times m \times l$	Point of a point cloud with geometry and intensity information
$\mathbf{p}^{x,y,z}$	$m \times m \times m$	Point of a point cloud with geometry information
\mathbf{p}^i	l	Point of a point cloud with intensity information
$\mathcal{N}_{\mathcal{P},r}(\mathbf{p}^{x,y,z})$	–	Local neighborhood of point cloud \mathcal{P} around point \mathbf{p} with radius r
\mathcal{X}	–	Set of vehicle poses
\mathbf{X}_v	$m \times m \times \text{rad}$	Vehicle pose in the UTM coordinate system
x_v	m	Easting of vehicle pose in the UTM coordinate system
y_v	m	Northing of vehicle pose in the UTM coordinate system
θ_v	rad	Heading of vehicle pose in the UTM coordinate system
D	–	Descriptor
\mathcal{D}	–	Set of descriptions
d	specific	Description
$\mathcal{F}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}^{3D}$	–	Feature Area with points \mathcal{P}_k and descriptions $\mathcal{D}_{\mathcal{P}_k}$
$\mathcal{F}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}$	–	2D Feature Area Representative of points \mathcal{P}_k and descriptions $\mathcal{D}_{\mathcal{P}_k}$
\mathcal{F}_O	–	On-board Feature Area
\mathcal{F}_M	–	Map Feature Area
$\mathcal{F}_{C,[t_1,t_n]}$	–	On-board Feature Area cluster for points in time t_1, \dots, t_n
$\mathcal{C}_{[t_1,t_n]}$	–	2D On-board Feature Area cluster representative for points in time t_1, \dots, t_n
\mathbf{R}	rad	Rotation matrix
\mathbf{t}	m	Translation vector
$\bar{\rho}$	specific	Arithmetic mean of random variable ρ
\mathbf{J}	specific	Jacobian matrix
\mathbf{C}	specific	Covariance matrix
$\mathbf{\Omega}$	specific	Information matrix
$x_{\text{UTM, VRF, SRF}}$	m	x -axis in UTM or ISO 8855 coordinate system
$y_{\text{UTM, VRF, SRF}}$	m	y -axis in UTM or ISO 8855 coordinate system
$z_{\text{UTM, VRF, SRF}}$	m	z -axis in UTM or ISO 8855 coordinate system

Part I.

**Context of LiDAR-Feature-based
Localization**

1. Introduction

This chapter motivates and introduces this thesis' subject of *LiDAR-Feature-based Localization* - a localization approach for automated vehicles using non-semantic patterns in the environment sampled by Light Detection And Ranging (LiDAR) sensors. The work puts an emphasis on practicality. This is reflected in the contributions, which are presented in Section 1.3. The introduction concludes with an overview of the thesis whose main part is divided into three content-related steps: i) the description of non-semantic patterns, ii) the detection of significant and persistent patterns, and iii) the localization using these significant and persistent detections together with their descriptions.

1.1. Motivating LiDAR-Feature-based Localization

Automated and autonomous driving has gained a lot of attention in the research community in the last years. The Self-Driving Systems (SDSs) have a high potential to reduce the number of road fatalities, which has been studied for example in the work of Vaa et al. [147]. 88% of all accidents in Germany are attributed to the misbehavior of the driver [124]. This includes turning errors, violation of right-of-way, too low distance to other road users, and unadjusted speed. This can be avoided by the introduction of SDSs. Additionally, automated and autonomous driving can increase the driver's productivity, e.g., by using the vehicle as a mobile office, and can provide an independent mobility, i.e., for people who are not allowed or able to drive [75].

Self-localization is an essential functionality for any automated vehicle. State-of-the-art SDSs need to know a global as well as a map-relative pose. For example, cooperative systems like vehicle-to-everything (V2X) systems exchange messages containing their poses to improve the road safety, e.g., with special vehicle warnings. In the context of V2X, the pose is given in a global coordinate system. This is important so that the receivers are able to interpret the messages' contents. However, the path planning task commonly relies on a priori knowledge in the form of map data because it requires further information that the vehicle might not be able to derive from its on-board sensor data. For instance, the road markings bounding the vehicle's lane might not be correctly recognized in unfavorable weather conditions. Therefore, such tasks also uses the vehicle's pose within a given map to be able to extract map information. The same applies to the navigation task at which the vehicle's navigation trajectory relies on map information and thus needs a map-relative pose as well.

Several popular localization solutions address the problem. As most maps for automated vehicles are globally referenced, standard input for localization are Global Navigation and Satellite Systems (GNSSs). Close to serial production GNSSs alone are in many cases not accurate enough. They suffer from strong multipath and blocked line of sight effects, especially in urban areas [2, 140, 141]. Even the poses of expensive Real-Time Kinematic (RTK) capable GNSSs with integrated Inertial Measurement Units (IMUs) are not reliable in regions with degraded or missing satellite reception, especially in long tunnels and parking garages or even in urban environments [72]. This result was confirmed by own evaluations with thorough test drives in Hamburg.

Hence, position information from close to production GNSSs is combined with localization based on additional sensor data, e.g., from cameras, LiDAR or Radio Detection And Ranging (RADAR) sensors. These sensors will most likely be available on automated or autonomous vehicles as they are necessary for detecting dynamic vehicles, anyway [77, 144]. One opportunity is offered by associating on-board

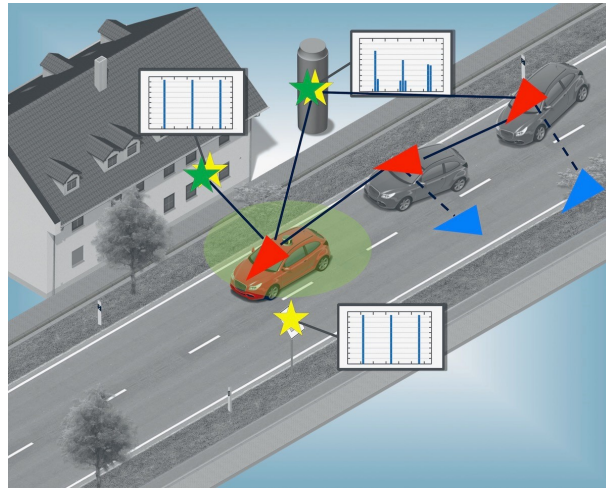


Figure 1.1: Concept of the *LiDAR-Feature-based Localization*. Scheme of the factor-graph-based localization using non-semantic detections, in this case, calculated with Fast Point Feature Histogram descriptors (white-blue bar charts) [110]. On-board detections (★) are matched with map elements (★) to estimate optimal present and past poses (▲) together with third-party pose information (▶), like GPS poses, and odometry information [55].

and offline saved sensor raw data, like LiDAR or RADAR point clouds [88, 118, 168]. Often, sparse, on-board point clouds are matched with dense and globally referenced map point clouds to achieve an accurate vehicle pose. This is for example done by minimizing the difference between on-board and map sensor data [9]. Raw data, especially dense map point clouds, require a large amount of memory and thus are hard to process. Another drawback of this approach is that raw data changes over time, hence the map is quickly outdated.

Other map-based localization solutions rely on static, semantic landmarks, like road signs or building corners, to reduce the computational effort and to make localization more robust [14, 24, 114, 160]. They estimate the transformation of on-board detected landmarks with respect to landmarks stored in a map. In this way, they calculate the vehicle pose in a map-relative coordinate system. By globally referencing the map, a global vehicle pose is computed simultaneously. However, this only works in predefined domains where these specified, semantic landmarks occur.

This thesis develops fundamentals of a localization solution based on non-semantic LiDAR features to be independent of semantic infrastructure elements, called *LiDAR-Feature-based Localization*. Based on camera data, such non-semantic approaches already exist [23, 148]. However, as automated vehicles will most likely be equipped with LiDAR sensors for the perception of the vehicle's environment, this thesis focuses on such sensors. They are particularly suited for the localization task as they measure the distance to the sampled environment and they are independent of lighting conditions. These properties distinguishes them from cameras, for which the application of features is widespread, e.g., SIFT and SURF [6, 79]. The baseline data of this approach are on-board detections of characteristic regions in LiDAR point clouds, called features, which are extracted without the use of semantic prior knowledge. Features are detected by looking for patterns in geometrical descriptions computed for every point of a point cloud. As this method is not based on prior knowledge, the description of every point of the LiDAR's point cloud is necessary. Next, the descriptions are used to detect prominent and persistent characteristics, the features, which benefit the localization task. This has to be done because not every pattern is suitable for localization. The feature observations are the input for the actual local-

ization algorithm, in this thesis a graph-based sliding window optimization, which is a state-of-the-art method for vehicle positioning [160]. Also, the matches of the on-board features with map features, GNSS measurements, and odometry are fed into the graph as nodes and vertices. The optimization of the graphs yields the estimation of the vehicle pose minimizing the errors between on-board observations and map data. The principles of the *LiDAR-Feature-based Localization* are visualized in Figure 1.1 and explained in more detail in the following section.

Within the thesis, the definition and implementation of each step of a method called *LiDAR-Feature-based Localization* is formulated. The thesis focuses on the analyzes of each step's practical benefit, especially the advantage for localization in terms of position accuracy, long-term robustness, and types of scenes in which features are of use. That means that this thesis focuses on examining whether descriptor algorithms and elements detected applying them are suitable for localization. It does not focus on developing a fully integrated localization solution, which should also consider other sources of positioning information like semantic landmarks. Finally, this thesis answers the following research question:

Research Problem:

Develop a procedure to detect robust non-semantic patterns in LiDAR sensor data with a high scenery coverage and demonstrate the usefulness of the detected patterns for vehicle localization.

1.2. Introducing Feature-based Localization

The *LiDAR-Feature-based Localization*, or more precisely the map-based localization for automated vehicles using LiDAR features, consists of three main steps.

LiDAR point clouds are the input for the first main step. The point clouds consist of the three-dimensional geometric information and further the normalized reflectivity of the sampled object. The main idea of the first step is to calculate a local description for every point of a preprocessed point cloud with a local neighborhood around that point with an algorithm, called descriptor. In this thesis, a sphere with the origin at the query point is used as local neighborhood. In the preprocessing step, the points are selected for which descriptions should be calculated. This is measured with the local neighborhood around a query point. If it includes enough points to characterize the three-dimensional structure a description calculation is performed. A descriptor function D is a function that maps the local neighborhood $\mathcal{N}_{\mathcal{P},r}(\mathbf{p})$ with radius $r \in \mathbb{R}_{>0}$ around a point \mathbf{p} to an element in \mathbb{R}^m , summarized in the following problem formulation of the first step:

$$D : \{\mathcal{N}_{\mathcal{P},r}(\mathbf{p}) \mid \mathcal{P} \subset \mathbb{R}^3 \times [0, 1], \mathbf{p} \in \mathbb{R}^3 \times [0, 1]\} \rightarrow \mathbb{R}^m. \quad (1.1)$$

Often, geometric properties of the sampling points within the point's local neighborhood, like the mean curvature or point distributions, are used as descriptor functions [35, 58, 110, 134]. But also, intensity information of the LiDAR point cloud is used to calculate descriptions [20, 42]. The outcome of the first main step of the method of this thesis are descriptor functions for production type sensors and real driving scenarios.

In the second main step, only those areas of the point cloud are extracted as input for the localization algorithm which are most useful for localization. These extractions are called features. As a description is assigned to every point of the preprocessed point cloud in the first step, the data size is automatically reduced to the subset of significant and persistent characteristics within the patterns' space. For that,

connected sets of points with similar descriptions are summarized in a cluster using descriptions and spatial information. The next step consists in choosing these sets of points based on a measure called Localization Information Gain (LIG). This measure is intended to capture the benefit for localization of each cluster to realize the selection in a non-semantic, automatic way. It captures the distinctiveness (low for features with descriptions of similar values and high for vectors with peaks), uniqueness (identifiability within a scene by its description), and spatial diversity (spatial distribution within a scene). The $LIG(\mathcal{F})$ for each feature \mathcal{F} is used to select the significant subset \mathcal{F} with significance level α and quantile $q_{1-\alpha}$, which leads to the overall problem formulation for the second step:

$$\mathcal{F} = \{\mathcal{F} \mid LIG(\mathcal{F}) \geq q_{1-\alpha}\}. \quad (1.2)$$

For on-board point clouds, all three values are considered. As this approach relies on map-data, the non-semantic features need to be extracted for the map as well. Usually, globally referenced, dense LiDAR point clouds form the basis for any map generation, e.g., maps for other localization concepts, path planning or navigation tasks. For the consistency of these maps, the map generation in this thesis is realized extracting map features from these point clouds. When extracting features from dense point clouds, only the distinctiveness is taken into account. All distinctive features are saved into the map. Thus, it includes all features which are considered helpful in the localization task besides the uniqueness and the spatial diversity. These measures are dependent on the current section of the environment whose consideration makes no sense in the map-generation process as this is a global display of the whole environment. The outcome of the second main step is a set of LiDAR features in the map that can be recognized in the vehicle.

The third main step comprises the localization algorithm. This includes several data association steps to make localization robust.

Firstly, the decision is made which features extracted from the on-board point cloud can be matched with features detected from previous process steps. Here, a distance measuring the feature's similarity is calculated for every feature extracted in each time step to every feature association group built in previous time steps. The local vehicle's odometry data is used to interpolate the position data of the features. In this way, dynamic objects changing their position and infrequently detected elements are filtered out.

Secondly, the decision is made, which of the associated features, the output of the previous step, can be matched with map features. This is done by transforming the local on-board features into a global coordinate system with an estimated pose of the localization computation of a previous time step. This is a significant advantage since, in this way, the possibility is created to deal with missing on-board feature detections, e.g., due to line-of-sight blockages, or features detected in the vehicle by mistake (parking cars' descriptions resembling walls). The map matching step is performed independently in every time step. It ensures that at the current time, a wrong map match can be dismissed, and missing map matches are still included in the vehicle pose estimation.

A graph-based optimization is applied to compute an optimal estimate of the vehicle state, including the vehicle pose, building a factor graph, among the odometry and other things, with the outcome of the previous steps: the clustered features and the map match factors. This graph is optimized, thus estimating the most likely vehicle state \mathbf{x}^* given the observations \mathbf{z} . This consists mainly in minimizing a sum of quadratic errors e_i given inaccuracies Ω_i :

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{x}|\mathbf{z}) \approx \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i e_i^T(\mathbf{x}, \mathbf{z}_i) \Omega_i e_i(\mathbf{x}, \mathbf{z}_i). \quad (1.3)$$

The outcome of the third main step is a global and map-relative pose of the vehicle, thus completing the *LiDAR-Feature-based Localization*.

During this thesis, not an Artificial Intelligence (AI)-based concept is pursued but a model-based approach. This is done due to the fact that the verification and validation has not yet been solved for AI-based processes and model-based methods are more comprehensible. Additionally, characteristic properties, such as geometry, can also be described in a conventional manner. As a third reason, this thesis' motivation is based on the fact that localization should not only be possible in predefined scenarios where, for example, semantic objects occur, but can be generalized. This is inconsistent with the use of training data and can lead to problems if the localization is used outside the domain of the training data.

1.3. Contributions

The research illustrated in this thesis provides the following contributions for localization of automated vehicles. The main contribution of this thesis is a comprehensive method for self-localization based on recognition of non-semantic features in LiDAR sensor data. The results of the thesis are based on a wide range of concepts, approaches and findings from the literature. These have been analyzed, improved and extended to be combined into a comprehensive method comprising the construction of feature maps from measurement data, online feature recognition from on-board sensor data, feature matching and advanced pose calculation employing error compensation. It has been validated on field data covering urban, rural and highway sceneries throughout one year. The validation established a high quality of the vehicle localization in many conditions. In detail, the contributions of the thesis are as follows:

Analyzes on the practicality of geometry-based descriptors and display of extensions and developments: The first contribution provides thorough analyzes of state-of-the-art 3D LiDAR descriptors and presents resulting extensions and new developments. Descriptors for point clouds already exist. However, these are often evaluated using very dense, synthetic point clouds, not using data from traffic sceneries. Therefore, the focus of the studies in this thesis lies on the potential of geometry-based descriptors for the localization task, especially in real-driving sceneries. This is measured with several criteria, e.g., distance and viewing angle independence. It can be shown that the descriptors specially designed for localization in this thesis improve the aspects which the state-of-the-art algorithms fail. The examinations, implemented extensions and new developments are provided in Section 3.2.

Development of a novel intensity-based descriptor algorithm - GRAIL: The second contribution introduces a novel descriptor algorithm describing local neighborhoods of LiDAR point clouds with intensity information. Only few descriptors focusing on the description of intensity information exist and they are not suitable for the application of this thesis. The gradients of build intensity shapes within the local neighborhoods are encoded into the local description, the GRAdients of Intensities as a Local descriptor (GRAIL). As the point clouds' intensities do not provide as much information compared to their 3D data, the extraction of non-semantic features is crucial for a working localization. Section 3.3 introduces this extraction method, including the GRAIL algorithm, and presents its potential for localization.

Development of a robust on-board feature extraction process: The third contribution consists of a feature extraction algorithm with on-board point clouds in a non-semantic way. There are very few extraction methods based on descriptors. For example, individual points, e.g., with a large surface curvature, are extracted. On the one hand, individual points are very difficult to robustly recognize for localization. On the other hand, points or objects that have no curvature, such as building walls, are also useful for localization, like for lateral positioning. Therefore, a

descriptor-based method has been developed in this thesis that is specially tailored for localization extracting features. A feature, or more precisely a Feature Area (FA), is a set of connected points with similar geometry-based descriptions rather than multiple single key points. The significant subset of FAs is the set of extractions, the input for localization. Their scene-dependent significance is measured with their benefit for the localization task without using class labels or other semantic information. The method is introduced and analyzed in more detail in Chapter 4.

Development of a automated map generation algorithm: The fourth contribution depicts a method for the automated generation of feature maps. Usually, maps are not only used for the localization task but also for the navigation and path planning problem. As the different tasks rely on different map information, the map is divided into several consistent map layers. In the context of the *LiDAR-Feature-based Localization*, a map layer containing non-semantic elements needs to be created. Commonly, the baseline data for any map generation are globally referenced, dense LiDAR point clouds. This data is used in the proposed feature map generation algorithm to extract scene-independent Feature Areas (FAs). Section 4.3 and Subsection 4.4.4 provides a detailed explanation creating non-semantic feature maps.

Integration of features into a graph-based localization: The fifth contribution presents an extension of a graph-based localization solution with features. The localization can be understood as a composition of a positioning and a Simultaneous Localization And Mapping (SLAM) approach just making use of the localization part of the SLAM algorithm. Familiar to the SLAM concept, the vehicle and feature positions are estimated while no update or generation of the map is implemented. The feature representations are fed into the graph whose optimization determines the vehicle and feature positions. For this, a robust data accumulation and the matching of the data with the map must be tailored to the data type, i.e., features, and implemented. The details of the feature representation are depicted in Chapter 4. The extensions of the feature integration into the graph-based localization and the localization accuracies are provided in Chapter 5.

Demonstration of the practical benefit of the *LiDAR-Feature-based Localization*: The sixth contribution presents the potential of the *LiDAR-Feature-based Localization*. The practical benefit is measured with several indicators. The persistence analysis with measurements of the same route over one year suggests that the *LiDAR-Feature-based Localization* works throughout the year, regardless of seasonal changes. The findings of the persistence analysis are explained in more detail in Subsection 4.5.2. Another indicator is the number of covered scenery types by the features, like rural, urban, or on highways. This shows the achieved independence of infrastructure elements with non-semantic elements. In Subsection 4.5.3, these findings are provided in more detail. The last indicator is the computational effort of the calculations. The generic non-semantic description and extraction of patterns are accompanied by a considerable computational effort. Nevertheless, localization should be real-time capable which is only possible with the concept of parallelization. This is depicted in Subsection 5.3.2.

1.4. Overview

The thesis is divided into three logical parts. The first part introduces the topic of *LiDAR-Feature-based Localization*. It includes in Chapter 2 a thorough literature review on the subjects of descriptors and detectors as well as localization based on non-semantic elements. Also, the chapter summarizes the mathematical basics which are applied in this thesis. The part concludes with derivations of the literature review for the context of localization for automated vehicles, i.e., the challenges of this thesis. Therefore, in the last section of this part, vehicle oriented research questions are defined, which form the basis for this research.

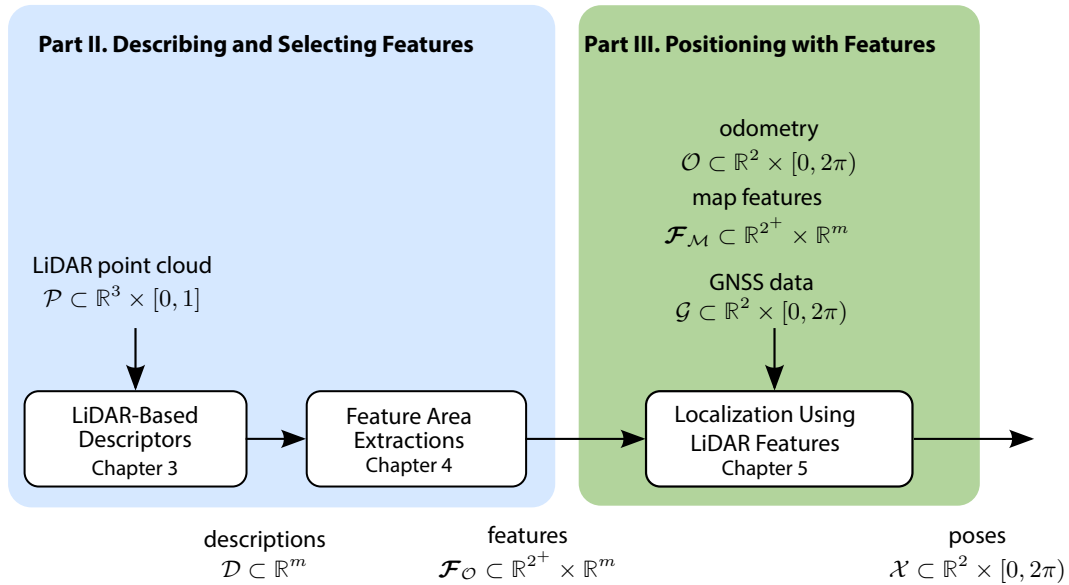


Figure 1.2: Overview of the thesis' central part

Hereafter, the central part of the thesis begins. It is illustrated in Figure 1.2, which additionally depicts the data flow of its main steps. It consists of the second part focusing in Chapter 3 on the descriptor algorithms. The algorithms assign every point of the point cloud \mathcal{P} a description, forming the set of descriptions of the whole point cloud \mathcal{D} . Several geometry- and intensity-based algorithms are explained in detail, extended or developed, and subsequently analyzed. The algorithms are analyzed with simulated, perfect point clouds based on LiDAR sensor models, post-processed, dense point clouds of the real environment, and sparse, on-board point clouds recorded with close-to-production type sensors. In Chapter 4, the detection of significant features using the geometry-based descriptions is outlined as only the most suitable objects should be used for localization. Features \mathcal{F} , or more precisely Feature Areas, aggregate a set of connected points to comprise the information of the point cloud. Their extraction should work with on-board point clouds resulting in on-board features \mathcal{F}_O as well as with dense point clouds, which are the input for the map generation process with map features \mathcal{F}_M . Both methods are presented in this chapter. The findings demonstrate that the proposed approach with areas as input for localization is more accurate than point-based localization on a real-world data set. Furthermore, this chapter answers parts of the practicability of the *LiDAR-Feature-based Localization*, i.e., the achieved scenery coverage and persistence of the approach. It can be seen that valid and persistent features including different type of objects are extracted with the detection method of this thesis on a highway, in urban, rural, suburb, and industrial areas.

The last part of the thesis displays the feasibility of the non-semantic localization approach. Chapter 5 depicts the graph-based sliding window localization with non-semantic features. In particular, the feature representations and their integration into the graph-based localization are introduced in this chapter. The localization further processes odometry \mathcal{O} , GNSS data \mathcal{G} , and map features \mathcal{F}_M . With the realization of this step, the map-based and global vehicle poses \mathcal{X} can be calculated in each time step. Afterward, the localization's accuracy is evaluated and demand of computational resource of the whole method.

Finally, Chapter 6 summarizes and discusses the thesis' outcomes. It also presents research problems going beyond the questions of this thesis.

2. Related Work and Basics

The thesis' main part is divided into three chapters, as introduced in Chapter 1. The ideas of each step, the concept of characterizing neighborhoods in LiDAR point clouds, detecting non-semantic elements, and positioning with these elements, have been a topic of research for quite some time. Thus, the results of these areas need to be considered when developing a holistic implementation for *LiDAR-Feature-based Localization*. Subsections 2.1.1 to 2.1.3 outline the relevant work independently for each of the three steps. Afterward, in Subsection 2.1.4, the review are taken as a basis to conclude research topics for this thesis. The last part of this chapter introduces existing technologies, methods, and mathematical fundamentals which will be referred to in this thesis.

2.1. Relevant Literature

The following provides a systematic literature review in the area of *LiDAR-Feature-based Localization*. The chapter begins with a discussion of the description of neighborhoods of LiDAR point clouds. After that, Subsection 2.1.2 outlines related extraction methods detecting outstanding elements with the geometry information of LiDAR point clouds, and Subsection 2.1.3 presents relevant literature on localization approaches relying on non-semantic elements.

2.1.1. LiDAR Descriptors

In this section, the related work on LiDAR descriptors is reviewed, the algorithms to characterize patterns, i.e., descriptions, within point clouds. The fundamental challenge in this field is to describe characteristic relations, like curvature or surface variation values, between sampling points in neighborhoods for every point of a LiDAR point cloud. Figure 2.1 summarizes the identified key LiDAR feature descriptor types and gives examples for each type.

There are two fundamentally different approaches to construct descriptor algorithms: AI- and model-based algorithms. This thesis focuses on handcrafted, i.e., model-based, descriptor algorithms to reduce the subjects examined in this thesis. An outlook of analyzes on AI-based descriptor algorithm can be found in appendix A. Therefore, this thesis and literature review gives a more detailed look into model-based descriptor algorithms. These methods can be divided into two categories: description- and detector-based methods. Description-based in comparison to detector-based techniques characterize every point of a point cloud, while detector-based methods select salient points based on simple attributes. The standard category is the description-based method, for which many algorithms have been developed.

AI-based Methods:

AI-based descriptor algorithms often use class labels during the training process. The most popular network processing 3D LiDAR data is the PointNet by *Qi et al.* [100]. The PointNet is designed to directly take point clouds without discretization, like voxelization or rendering, as input and output class labels. The PointNet architecture consists of two core building blocks, the Transformation Networks (T-Nets) and the symmetric function. The T-Nets, in this case Spatial Transformation Network (STN) [127], are

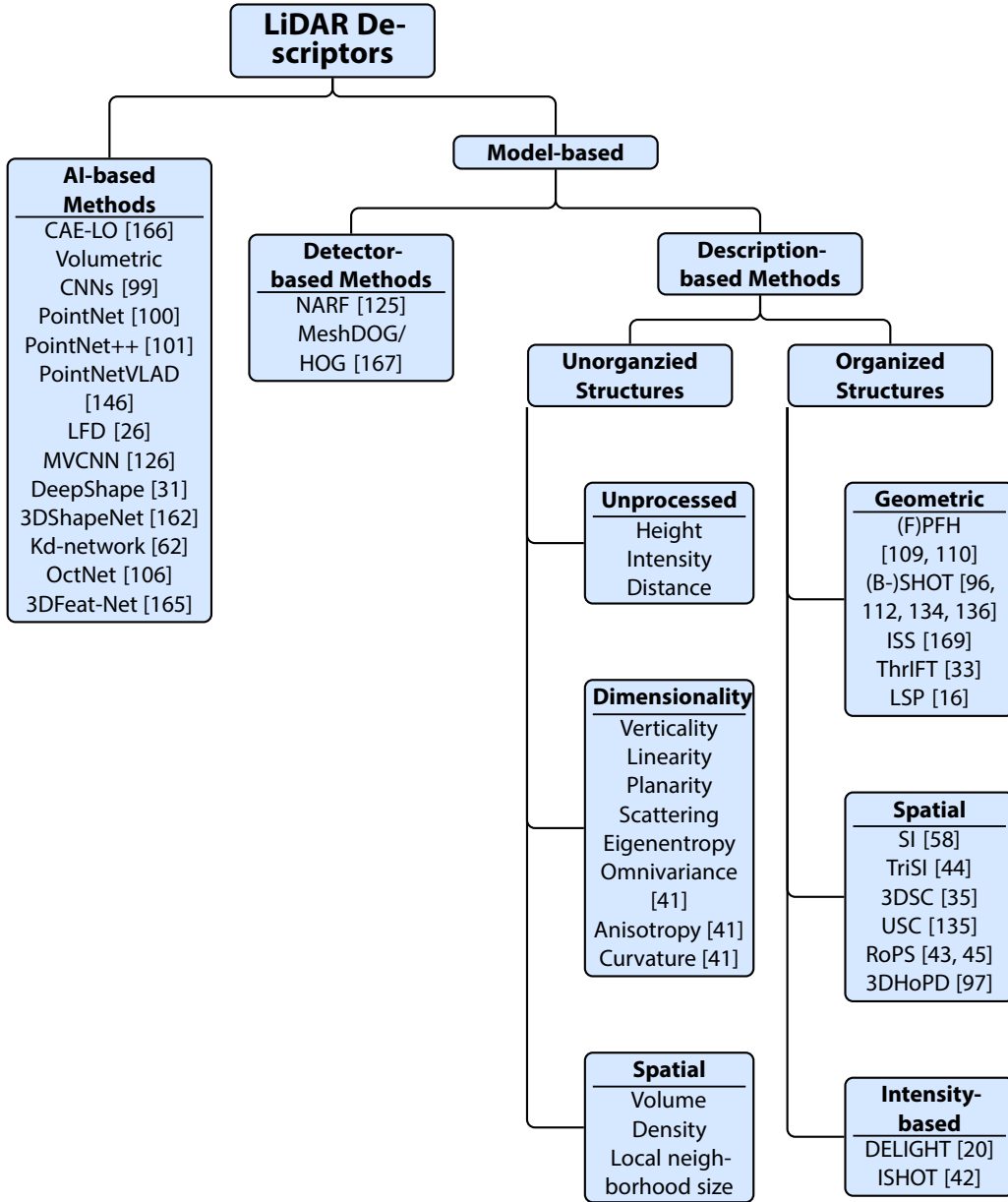


Figure 2.1: Overview of 3D feature descriptor types with some examples

used to deal with spatial invariances. Part of the network architecture of the PointNet, the symmetric function realized through the max-pooling process, is the calculation of a feature vector used for the classification of the point cloud. This feature vector can be interpreted as a description vector for the entire input. The vector is used to learn the classification using Multi-Layer Perception (MLP). As the MLP is designed to learn the feature vector from each point independently and disregards relations between points, the PointNet++ was developed by *Qi et al.* to represent local feature vectors of neighboring points [101]. Furthermore, there are several works based on the research of *Qi et al.*, like [146, 165], using semantic class labels and calculating within the network architecture description vectors [26, 31, 62, 99, 106, 126, 162].

The training process of the AI-based methods described above relies on semantic class labels. Applying them to the context of this work, self-localization for automated vehicles, the learned descriptions only work in scenes containing those object categories. Therefore, the application of semantic class labels to compute description vectors is unfavorable because the approach of this thesis is to detect non-semantic elements circumventing the scene dependence. In contrast to this, a procedure by *Yin et al.* learns local feature descriptions without class labels [166]. This fully unsupervised method is based on Convolutional Auto-Encoders (CAEs) to detect key points and determine their description vectors. As input for the CAE local voxel patches are generated with centers on the interest points. The CAE, with pooling layers and fully connected layers, compresses the voxel to represent the description vector. The Convolutional Auto-Encoder based LiDAR Odometry (CAE-LO) is an example for artificial detector-based methods.

Model-based Methods:

Furthermore, there exist many hand-crafted descriptor algorithms. These hand-crafted algorithms depend on models and therefore are called model-based methods. The literature review on them is presented in the following.

Detector-based Methods: In the following, two further detector-based methods are listed, which are hand-crafted. The first detector-based algorithm by *Steder et al.*, namely Normal Aligned Radial Feature (NARF), is based on depth images generated from point clouds [125]. The main idea of the NARF algorithm is to detect key points from the point cloud and describe them using depth information. A key point is defined as a point of the point cloud i) which is independent of the view angle, ii) in whose local neighborhood great depth changes occur, and iii) whose dominant direction of the depth changes is significant. Subsequently, the descriptions values are determined for every extracted key point. Therefore, a normal aligned distance patch is generated around each key point. Within the patch, depth changes along multiple orientations with the center at the patch center, forming a star-like pattern, are considered for the description computation - every orientation results in one description value. The higher the depth changes along an orientation are, the higher the description value gets. In order to achieve rotational invariance, the highest value, i.e., depth change in one orientation, can become the first entry of the description vector maintaining the order of the orientations.

The second detector-based algorithm is by *Zaharescu et al.* [167]. The descriptor algorithm Mesh Histogram of Oriented Gradients (MeshHOG) is performed on key points detected with the Mesh Difference Of Gaussians (MeshDOG) extraction method. A mesh is defined as a discrete surface domain on which the MeshHOG is performed. The MeshHOG is a generalization of the Difference Of Gaussians (DOG)¹ used to construct a discrete Laplacian operator. It enables a representation of scalar functions over multiple scales, in this case, convolution with a discrete Laplacian operator, and the detection of key points as local extrema. The MeshHOG is a generalization of the descriptor algorithm

¹The DOG is a subtraction of an original image smoothed with a Gaussian kernel from another, less smoothed variant of the original.

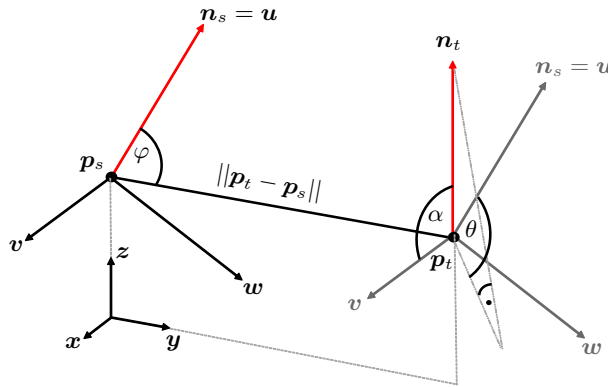


Figure 2.2: Darboux frame (u, v, w) between a point pair (p_s, p_t) for the calculation of the FPFH angles $\theta, \alpha,$ and φ - based on [110]

Histogram of Oriented Gradients (HOG) for 2D images by *Dalal and Triggs* [22]. For it, a local, normal aligned coordinate system is computed. Gradient vectors determined with the Laplacian for each of the three planes of the coordinate system are the input for the HOG algorithm yielding description values of the MeshHOG method.

Description-based Methods: Since detector-based methods reduce the input dimension for the description calculation, they are faster to process than handling the whole point cloud as in description-based procedures. However, detector-based methods extract single key points which are not robust for a self-localization, especially concerning sparse LiDAR point clouds from close-to-production sensors. Additionally, elements used for localization should be persistent, which is hard to examine with single points or general characteristics as introduced before. Thus, the description-based method is pursued in this thesis. A distinction should be made between organized and unorganized structured descriptors. While unorganized structured descriptors directly capture the local neighborhood's characteristics, e.g., mean height, mean intensity or scattering, organized structured descriptors arrange the characteristics in a specific order, e.g., histograms or signatures. Several popular descriptors are explained in the following, some of them in more detail as they are used for examinations later in the thesis.

Unorganized Structures: Unorganized structured descriptors are partitioned into three groups: i) spatial, ii) dimensionality, and iii) unprocessed descriptors. *Spatial descriptors* capture the points' spatial expansion within a local neighborhood, like volume, point density, or local neighborhood size, which are easy to implement. *Dimensionality descriptors* measure the salient direction in space within a local neighborhood, like linearity, planarity or scattering. Often, the computation of those descriptions depends on eigenvalue calculations, e.g., by *Gross and Thoennessen, Han et al., Vandapel et al.* [41, 47, 149]. The three eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ are obtained by a Principal Component Analysis (PCA) with the local neighborhood around the query point. The eigenvalues' sum is normalized to one. Resultant, the linearity l , planarity p , scattering s , anisotropy a , eigenentropy e , omnivariance o , and the surface variation v_s can be determined:

$$l = \frac{\lambda_1 - \lambda_2}{\lambda_1}, p = \frac{\lambda_2 - \lambda_3}{\lambda_1}, s = \frac{\lambda_3}{\lambda_1}, a = \frac{\lambda_1 - \lambda_3}{\lambda_1}, \quad (2.1)$$

$$e = -\lambda_1 \ln(\lambda_1) - \lambda_2 \ln(\lambda_2) - \lambda_3 \ln(\lambda_3), o = 3 \cdot \sqrt[3]{\lambda_1 \lambda_2 \lambda_3}, v_s = 3 \cdot \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}.$$

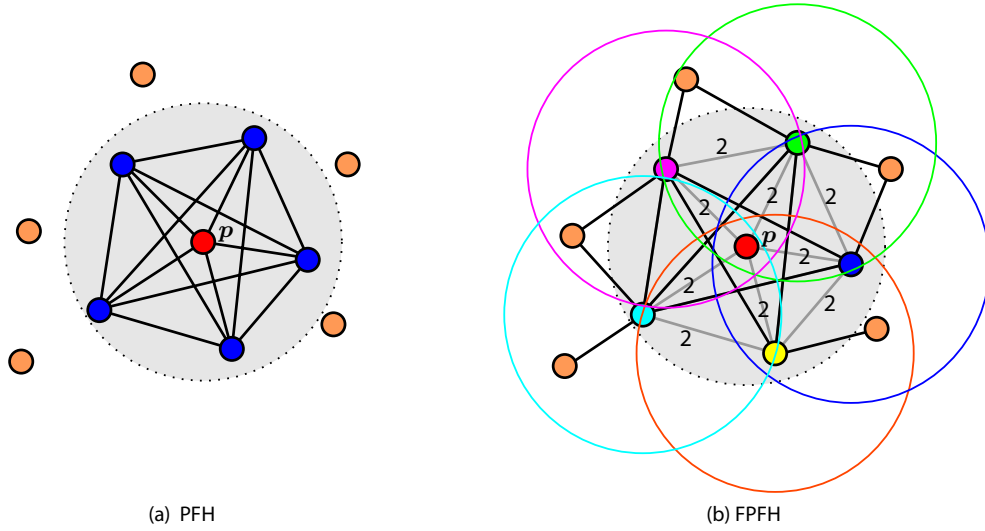


Figure 2.3: Relations between sampling points in the local neighborhood (gray) with radius r around query point p (red) for the description calculation of the PFH and FPFH. In the case of the PFH, relations between every point pair of the local neighborhood are taken into account. In the case of the FPFH, not every point pair is considered but some with double weight (2 on point connection) and even some outside the local neighborhood (orange) - based on [108–110].

All descriptions of Equation 2.1 take on a value in $[0, 1]$, due to the normalization of the eigenvalues. *Unprocessed descriptors* directly characterize the points' coordinates, like height or distance, or intensities. All of these unorganized structures are easy to process and implement. However, they are dependent on the local neighborhoods resolution, especially for sparse on-board point clouds, and little descriptive.

Organized Structures: Organized structured descriptors can be divided into three categories: i) geometric, ii) spatial, and iii) intensity-based. *Geometric descriptors* capture the 3D information of the point cloud's local neighborhoods, like surface variation values or curvature. A histogram-based descriptor algorithm is the PFH [108, 109] and its further development FPFH [110] by *Rusu et al.*, which are part of the Point Cloud Library (PCL) [107]. Both describe the mean curvature around the query point within its local neighborhood using surface normals. Using surface normals serves to achieve independence from view angles.

The calculation of the PFH and the FPFH for a query point of a point cloud are very similar. It starts with the formation of point pairs within the local neighborhood around the query point. For every point pair, the point of the point pair is defined as source point p_s whose normal n_s and connecting line $\|p_t - p_s\|$ form the smaller angle and the other is the target point p_t . These points are used to build the Darboux frame. The Darboux frame can be seen in Figure 2.2. The axes of the Darboux frame $u \perp v \perp w$ with their origin in p_s are defined as

$$u = n_s, v = u \times (p_t - p_s), w = u \times v. \quad (2.2)$$

The Darboux frame is used to calculate three angles θ , α , and φ describing the point pair's curvature:

$$\begin{aligned}\theta &= \operatorname{atan}\left(\frac{\mathbf{u} \cdot \mathbf{n}_t}{\mathbf{w} \cdot \mathbf{n}_t}\right) \\ \alpha &= \operatorname{acos}(\mathbf{v} \cdot \mathbf{n}_t) \\ \varphi &= \operatorname{acos}\left(\mathbf{u} \cdot \frac{\mathbf{p}_t - \mathbf{p}_s}{\|\mathbf{p}_t - \mathbf{p}_s\|}\right).\end{aligned}\tag{2.3}$$

This calculation results in 3-tuples for every point pair of the local neighborhood.

For the PFH descriptor, this is computed for every point pair within the local neighborhood, refer to 2.3a. Its description vector is formed by dividing the three angles into five intervals, respectively. Binning every combination into a histogram yields a $5^3 = 125$ -dimensional vector. For the FPFH descriptor, the angles of Equation 2.3 are determined between the query point and its neighbors and between their neighbors, refer to Figure 2.3b. All of these 3-tuples are binned independently into a histogram, which is normalized to 100%. This histogram is called Simplified Point Feature Histogram (SPFH).

The SPFH is calculated not only for the local neighborhood of the query point but for every local neighborhood of all points in the query point's neighborhood. The final FPFH is determined from all k SPFHs of the query point's \mathbf{p}_i local neighborhood:

$$\operatorname{FPFH}(\mathbf{p}_i) = \operatorname{SPFH}(\mathbf{p}_i) + \frac{1}{k} \sum_{j=1}^k \frac{1}{w_j} \cdot \operatorname{SPFH}(\mathbf{p}_j),\tag{2.4}$$

with weight $w_j = \|\mathbf{p}_i - \mathbf{p}_j\|$, which represents the distance between the query point \mathbf{p}_i and each unequal neighbor point \mathbf{p}_j .

Similar to the FPFH approach, the Signatures of Histograms of Orientations (SHOT) descriptor introduced by Tombari *et al.* depicts relations of surface normals in a subdivided local neighborhood, here a sphere [112, 134, 136]. It combines geometric information with spatial information. Since the describing part of the descriptor consists of the determination of normal differences, it is assigned to the geometric descriptor category. A robust Local Reference Frame (LRF) computation based on the PCA is performed for every point of the point cloud. This LRF is robustified by weighting the local neighborhood points depending on their distance to the query point. Next, the local neighborhood is divided using an isotropic spherical grid, as shown in Figure 2.4a. Therefore, two different local neighborhood radii are used. Both spheres are divided in their middle and the resulting parts are further divided along the azimuth angle. For each part of the grid, the differences between the normals of the points and the query point are accumulated into histogram bins. The final SHOT is the concatenation of every histogram of each partition. Prakhya *et al.* introduced a variation of the SHOT, called Binary Signatures of Histograms of Orientations (B-SHOT), to decrease the computation time and the memory capacity [96]. Each value of the SHOT is replaced with binaries encoding every successive four values of the SHOT.

Another geometric descriptor, namely the Intrinsic Shape Signatures (ISS), was devised by Zhong [169]. This descriptor, like the SHOT, unites geometric information with spatial information but is categorized as a geometric descriptor due to its characterization focus. First, a LRF ($\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_1 \times \mathbf{e}_2$) is determined for a query point with a spherical local neighborhood around it, where \mathbf{e}_i are the eigenvectors computed with a PCA. Then, an octahedron is used to create a spherical grid of the angular space with several partitions. A sample grid is visualized in 2.4b. Finally, the weighted sum of points in each partition is binned into a 3D histogram, which is the definite description.

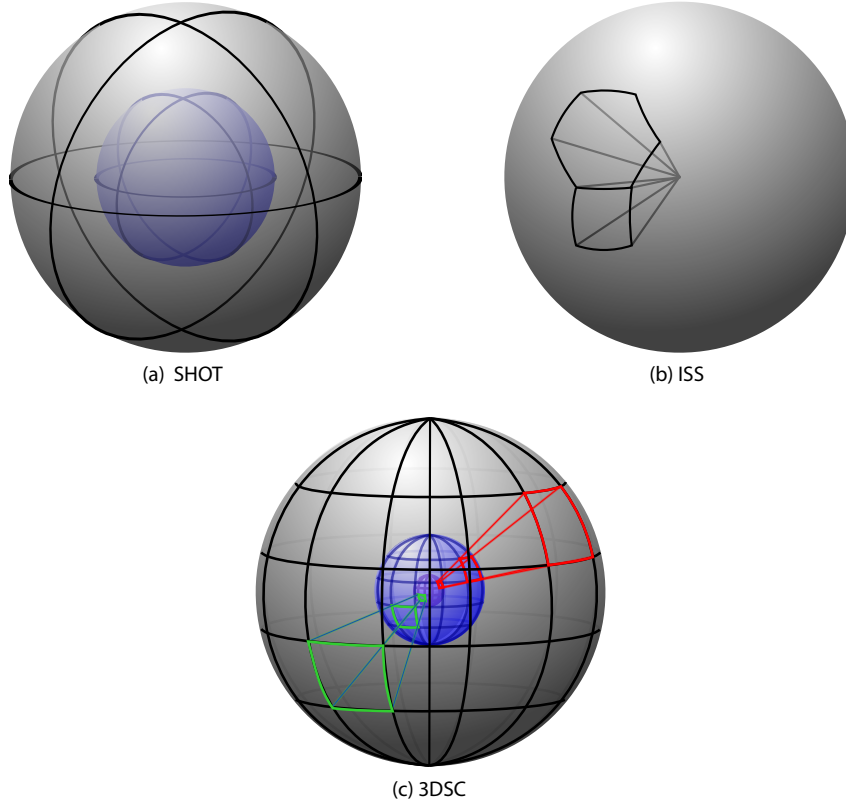


Figure 2.4: Local neighborhood partitions of different description calculations - based on [35, 134, 169]

Based on the popular work on the object recognition in 2D images, Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Feature (SURF) of *Lowe* and *Bay et al.*, the ThrIFT descriptor was developed by *Flint et al.* [6, 33, 79]. This descriptor adapts the idea of SIFT and SURF to work with 3D point clouds considering orientation information. Two normals $\mathbf{n}_{\text{small}}$ and $\mathbf{n}_{\text{large}}$ are determined from the least squares' plane windows, a small and a large-scaled window, for each query point and point within the query points' local neighborhood. The angle β between those two normals is binned into a histogram forming the description:

$$\beta = \arccos \left(\frac{\mathbf{n}_{\text{small}} \cdot \mathbf{n}_{\text{large}}}{\|\mathbf{n}_{\text{small}}\| \cdot \|\mathbf{n}_{\text{large}}\|} \right). \quad (2.5)$$

Chen and Bhanu introduced a descriptor called Local Surface Patches (LSP) [16], which is also based on the comparison of normals. It describes local neighborhoods around query points by binning the shape index values of neighboring points and the normal angles between the query point and the neighboring points into a histogram.

Spatial descriptors, as in unorganized structured descriptors, measure the spatial expansion within a local neighborhood while binning it into ordered systems, like intervals. The most popular local 3D descriptor method is the Spin Images (SI) algorithm by *Johnson and Hebert* [58]. The SI algorithm characterizes the relations of neighboring points with two distances. The points' distance on their local

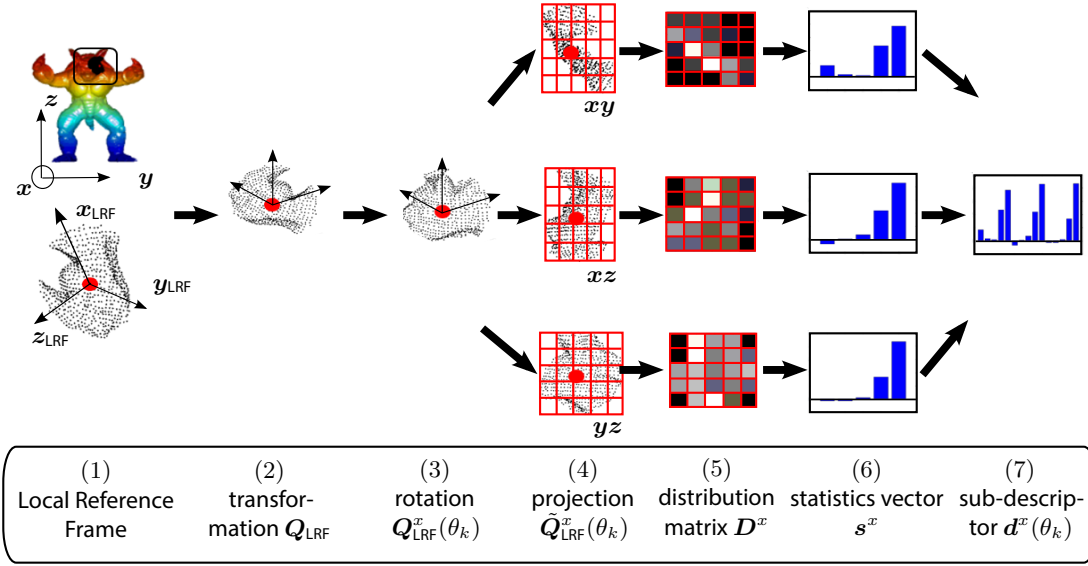


Figure 2.5: Scheme of RoPS description construction for one sample rotation - based on [43]

tangential plane dist_t and the orthogonal distance dist_n to their neighbor points p_n are calculated for every point within their local neighborhood:

$$\text{dist}_t = n_q \cdot (p_q - p_n), \text{dist}_n = \sqrt{\|p_q - p_n\|^2 - \text{dist}_t^2}, \quad (2.6)$$

with n_q as the normal of the query point p_q . For every neighboring point, these values are binned into a histogram resulting in the SI description vector. A variation of the SI descriptor is the Tri-Spin-Image (TriSI) descriptor by *Guo et al.* [44]. It builds three SIs by rotating the SIs around each axis of a LRF. The final TriSI is generated with the concatenation of the three SI descriptions.

Frome et al. propose a descriptor algorithm called 3DSC [35]. The local neighborhood around a point of the point cloud is built using a spherical support region. The north pole of the sphere is pointing in the direction of the surface normal. Then, the sphere is equally divided into partitions with fixed azimuth and elevation and logarithmic spacing radial dimension. Finally, the 3DSC description is calculated as the weighted sum of the number of points falling into the partitions. The approach called Unique Shape Context (USC) by *Tombari et al.* [135] extends the idea of the 3DSC. It computes the spherical histogram in the same way while adding a unique, local reference frame with an eigenvector decomposition weighted covariance matrix.

The descriptor RoPS [43, 45] by *Guo et al.* is based on a similar idea. A local reference frame is used to describe the distribution (central moments and eigenentropy) with five statistics while rotating the local neighborhood around all three axes of the LRF. In Figure 2.5 all seven steps of the RoPS calculation are presented. Here, the points of the local neighborhood, a sphere, are called local point cloud $Q = \{q_1, \dots, q_{n+1}\}$. (1) First, the LRF is built with a PCA and the local point cloud is (2) transformed into the LRF Q_{LRF} . (3) For every angle θ_k , the transformed point cloud is rotated around the x_{LRF} axis yielding the rotated point cloud $Q_{\text{LRF}}^x(\theta_k)$. (4) The discretized projections $\tilde{Q}_{\text{LRF}}^x(\theta_k)$ of the rotated point cloud into all three planes of the LRF are (5) the input for the calculation of the distribution matrix D . The distribution is characterized as the normalized number of points within each discretization cell.

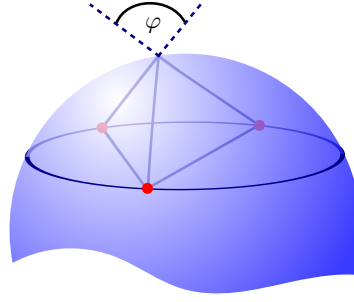


Figure 2.6: Specification of the simplex angle - based on [50]. The simplex angle φ is defined as the cutting angle between the two triangles (blue lines) built by three neighbor points (red) and a sphere (blue) formed by those neighbor points.

(6) The four central moments μ_{11} , μ_{12} , μ_{21} , and μ_{22} , and the Shannon entropy e by [120] summarize the distribution matrix D^x :

$$\mu_{mn} = \sum_{k=1}^L \sum_{s=1}^L (k - \bar{k})^m (s - \bar{s})^n D^x(k, s), \quad e = - \sum_{k=1}^L \sum_{s=1}^L D^x(k, s) \log(D^x(k, s)), \quad (2.7)$$

$$\text{with } \bar{k} = \sum_{k=1}^L \sum_{s=1}^L k D^x(k, s), \quad \bar{s} = \sum_{k=1}^L \sum_{s=1}^L s D^x(k, s).$$

This yields for the three distribution matrices of each plane a 5D statistics vector s^x , which are concatenated (7). Step (3) to (7) are repeated for rotation with every angle θ_k around each axis what results in a $3 \cdot k \cdot 5$ -dimensional description $d(\theta_k)$ concatenating the statistics vectors.

The 3D Histogram of Point Distribution (3DHoPD) descriptor by *Prakhya et al.* depicts depth information of the point cloud [97]. First, the local neighborhood is transformed into a LRF. Subsequently, the range along the x -axis is partitioned into intervals and binned into a histogram. This partitioning is repeated for the y - and z -axes and merged into the final description.

Intensity-based descriptors process the local neighborhoods' reflectivity instead of 3D information as before. It has received some attention in recent years. *Barfoot et al.* [5] generate intensity images from high-resolution LiDAR data recorded with a ILRIS3D, extract SURF descriptions, and perform feature-based localization in dark environments. In contrast to that, *Cop et al.* [20] introduce the DDescriptor of LiDAR Intensities as a Group of HisTograms (DELIGHT). It is a global descriptor computing one vector for the point cloud by counting the differences of raw reflectivity returns of subdivisions. The Intensity Signatures of Histograms of OriEnTations (ISHOT) descriptor by *Guo et al.* [42] refined the idea of the DELIGHT descriptor. Instead of counting the differences of intensity returns, the ISHOT method bins the direct intensity return of a local neighborhood into a histogram. Without any preprocessing step, the last two descriptors are dependent on different distances and point cloud densities.

2.1.2. Extraction of Non-Semantic Elements

This section provides an overview of the relevant literature on the detection of non-semantic elements in 3D LiDAR data. The research area handles the detection of prominent and persistent descriptions along with their precise position within a point cloud. The literature review is based on previous research of several survey papers [39, 46, 90, 111, 137, 143] - Figure 2.7 structures the different extraction methods.

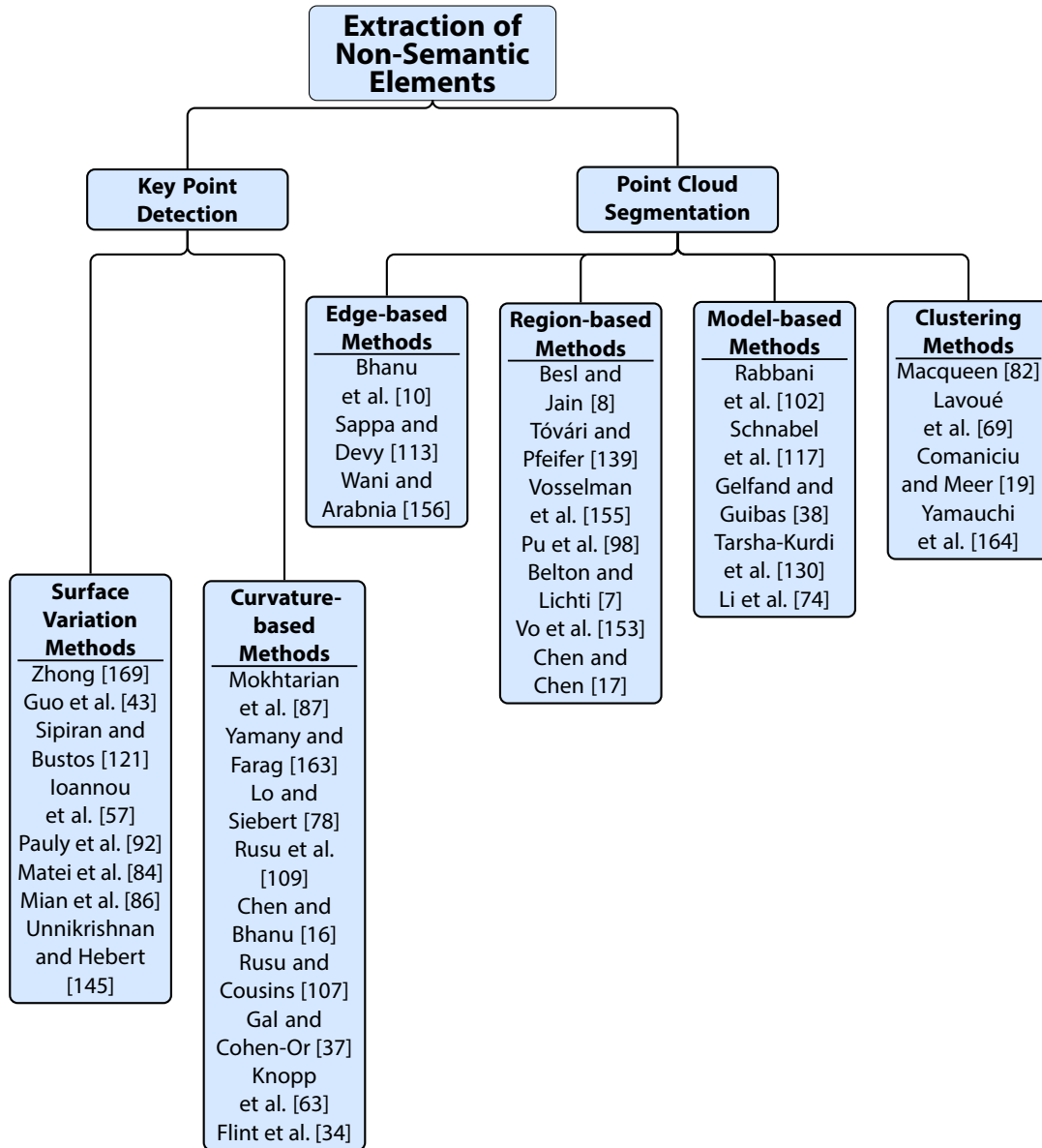


Figure 2.7: Overview of extraction of non-semantic elements from 3D point clouds with some examples.

The extraction of non-semantic elements can be classified into two categories: key point detection and point cloud segmentation. Key point detection is the process of extracting local surfaces around specific, distinct key points of 3D point clouds. Often, key point detection is applied in the field of object recognition [46]. Point cloud segmentation is the problem of dividing a 3D point cloud into multiple homogeneous areas consisting of points with similar characteristics [90]. Here, the key points' distinctiveness and the segments' characteristics are measured with mostly simple descriptors or sometimes with complex descriptors described in Subsection 2.1.1. In the following, the review of both research areas is presented.

Key Point Detection:

Non-semantic elements in 3D LiDAR data are often key points extracted based on prominent description-like characteristics. For instance, often the curvature and surface variation values within a query points local neighborhood are used to detect prominent points in a non-semantic way. In this thesis, they are divided into two categories: curvature-based and surface variation methods. Both of them are presented hereafter.

Curvature-based Methods: Curvature values can be used as distinctiveness measures to detect prominent key points in multiple ways. *Mokhtarian et al.* [87] determine the Gaussian and mean curvature around a point using a fixed neighborhood. Points whose curvature values exceed a minimum threshold within their neighborhood are declared as key points. The curvature is depicted with the simplex angle φ , defined by *Hebert et al.* [50], in the work of *Yamany and Farag* [163] to detect key points. This angle measures the angular variation between neighboring points. It is called simplex angle because it is the angle between two 2-simplices, triangles, of a 3-simplex, tetrahedron, built with two neighbor points, refer to Figure 2.6 for a visualization. A point is specified as key point if the simplex angle satisfies the constraint

$$|\sin(\varphi)| \geq t \quad (2.8)$$

based on a global threshold t .

Furthermore, several works apply elaborate descriptors, refer to Subsection 2.1.1, to extract key points based on fixed thresholds [16, 34, 78, 109]. In the work of *Lo and Siebert* [78], an enhanced version of the SIFT algorithm by *Lowe* [80], called 2.5D SIFT, is introduced. They compute multiple scaled depth images within the point cloud by smoothing the depth images with a Gaussian kernel and calculating DOGs. These depth images are the input for comparing the ratio of the points' principal curvatures to a fixed threshold. Points whose ratio is higher than the threshold are extracted as key points. In the work of *Flint et al.* [34], they adopted their ThrIFT descriptor for the key point extraction task. The adopted ThrIFT is calculated by subtracting the angles between the normal of the query point and these of the neighboring points. These angles are binned into a weighted histogram. All points whose differences are larger than a threshold are assigned as key points. Moreover, *Rusu et al.* [109] select points whose PFH descriptions, refer to Subsection 2.1.1, differ by a chosen factor from the standard deviation to the pre-computed mean description of the considered data set. Consequently, with this scene-dependent threshold, they take global, not local variations into account.

However, the performance of those key point detection methods highly depends on the choice of an appropriate threshold. *Chen and Bhanu* [16] overcome this issue by defining LSP with shape index values for every point, which characterize shape variations around each point, refer to Subsection 2.1.1. A key point is identified by a local optimum of the shape index value within a neighborhood. These characteristics lead to spatially distributed key points but is sensitive to noise. Another adaption of the SIFT algorithm, namely 3D-SIFT, was developed as a part of the Point Cloud Library (PCL) [107] to detect key points. First, the Gaussian scale-space is calculated for a local neighborhood of the point cloud by downsampling the neighborhood with differently sized voxel grid filters and smoothing the

downsampled points. The smoothing is realized with a weighted average of the neighbor points' principal curvatures. The DOG method yields local optima within the neighborhood whose corresponding points are declared as key points. Afterward, all key points in regions with low curvature values are rejected. *Gal and Cohen-Or* [37] also detect key points without the use of thresholds by computing a linear combination of the sum of curvatures of neighboring points and the variance of the curvature values within a neighborhood. This linear combination is called saliency grade. The resulting saliency grades for every point are used for the key point extraction by a region growing algorithm. Furthermore, the idea of *Knopp et al.* [63], namely 3D SURF, is to voxelize the local neighborhood of the point cloud and compute second-order derivatives at each voxel with increasing standard deviations. Local optima within a saliency measure based on the Hessian matrix over all voxels and deviations are used to define key points.

Surface Variations Methods: Also other surface variation values can be used to omit key points from 3D LiDAR data. The work by *Zhong* [169] and by *Guo et al.* [43] select key points used as input for their description calculation of ISS and RoPS, respectively, refer to Subsection 2.1.1. Their extraction is based on the ratio of combinations of the eigenvalues' magnitudes within a fixed neighborhood. The eigenvalues of every local neighborhood are computed with a PCA and sorted in a decreasing order $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. *Zhong* keep those points of the point cloud whose eigenvalues satisfy the following conditions:

$$\frac{\lambda_2}{\lambda_3} < t_{2,3} \wedge \frac{\lambda_1}{\lambda_2} < t_{1,2}, \quad (2.9)$$

while respecting predefined thresholds $t_{2,3}$ and $t_{1,2}$. *Guo et al.* choose those points as key points whose local neighborhoods' eigenvalues fulfill

$$\frac{\lambda_3}{\lambda_2} > t, \quad (2.10)$$

for some fixed threshold t . There are further approaches, e.g., presented by *Pauly et al.* [92], *Matei et al.* [84] and the Key Point Quality (KPQ) by *Mian et al.* [86], that use eigenvalues to derive a surface variation measure for key point detection. *Pauly et al.* apply the surface variation value v_s of Equation 2.1. Local maxima within the smoothed surface variation space are estimated to extract the key points. *Matei et al.* only use the smallest eigenvalue λ_3 to measure the surface variation and assign those points with a large λ_3 as key points. First, *Mian et al.* rotate the local neighborhood points around a query point such that the normal, computed with a PCA, is aligned with the z -axis. Here, the ratio of the two largest eigenvalues $\lambda_1 > \lambda_2$ determined with the PCA describes the surface variation. Points who meet the following condition

$$\frac{\lambda_1}{\lambda_2} > t, \quad (2.11)$$

where t is a fixed threshold, are kept as key points.

Sipiran and Bustos [121] proposed a detector called Harris 3D, an extension of the Harris Corner Detector [48]. First, the neighboring points for every point of the point cloud are translated to the neighborhoods' centroids. Then, they are rotated into the best fitting plane such that the z -axis of the plane is aligned to the normal of that neighborhood. The normal is assumed as the smallest eigenvalue computed with a PCA. Subsequently, the neighboring points are translated such that the query point is in the center of the xy -plane. A quadratic surface is fit to the translated points with the least square method of the paraboloid form

$$f(x, y) = \left[\frac{a_1}{2}, a_2, \frac{a_3}{2}, a_4, a_5, a_6 \right]^T [x^2, xy, y^2, x, y, 1]. \quad (2.12)$$

Hereafter, the derivatives of the quadratic surface are calculated with a continuous Gaussian function to formulate the autocorrelation matrix \mathbf{E} . The surface variation value, namely Harris operator value, related to the neighborhood structure of the query point is calculated as in the Harris Corner Detector

$$\det(\mathbf{E}) - k (\text{tr}(\mathbf{E}))^2, \quad (2.13)$$

where $\det(\cdot)$ and $\text{tr}(\cdot)$ represent the determinant and the trace of a matrix, respectively, and k is a constant factor that needs to be determined experimentally. The query points with the highest Harris operator value are extracted as key points.

Ioannou et al. [57] model an operator, called multi-scale Difference Of Normal (DON), characterizing the surface variation of different local neighborhood radii. The normals are calculated with a PCA using different sized local neighborhood radii around the same query point. One normal is negated if the angle between two normals of two different radii is greater than $\frac{\pi}{2}$ to maintain a consistent direction of the normals throughout the normal calculation of different radii. Points with a small DON values, i.e., less surface variation, are discarded. Furthermore, *Unnikrishnan and Hebert* [145] proposed an integral operator $I_{\text{op}}(\mathbf{p}, r)$ to measure the surface variation around point \mathbf{p} with local neighborhood radius r and detect key points based on it. The integral operator is determined by displacing the query point along its normal and in proportion to its curvature, i.e., the mean curvature within its local neighborhood. Next, the surface variation is defined as

$$\frac{2 \|\mathbf{p} - I_{\text{op}}(\mathbf{p}, r)\|}{r} \exp\left(-\frac{2 \|\mathbf{p} - I_{\text{op}}(\mathbf{p}, r)\|}{r}\right) \quad (2.14)$$

to keep those points whose surface variation is larger than a threshold.

Point Cloud Segmentation:

Another approach detecting prominent characteristics in 3D point clouds sampled with LiDAR sensors is the point cloud segmentation. Point cloud segmentation is the problem of dividing the point cloud into non-semantic elements, i.e., areas with small changes in their characteristics. The 3D point cloud segmentation procedures can be grouped into four categories: edge-based, region-based, model-based, and clustering methods, refer to Figure 2.7. All of them are presented in the following.

Edge-based Methods: Edge-based methods divide the point cloud into different regions in two central steps: i) identifying edges to contour the borders of multiple areas, and ii) grouping points inside the edges to extract the segments. The edges are detected by local variations in the points' surface characteristics, similar to simple descriptors Subsection 2.1.1. Often, characteristics like gradients, normals, or principal curvatures are taken into account.

Bhanu et al. [10] introduced three edge detection methods. The first approach is to calculate depth gradients in the point cloud to obtain the magnitude and direction of an edge. If the edge magnitude of a query point exceeds some threshold or the size of its two neighbors with the same edge direction, the query point belongs to an edge. They connect points belonging to the same edge with the edge magnitude and direction. The second approach is to fit straight lines into local neighborhoods of the point cloud. For this, the unit direction vector for each point of the point cloud is calculated. If two direction vectors point in the opposite direction, the corresponding query points lie on a straight line. If there are at least two straight lines within a local neighborhood threshold, the neighborhood's query point is no edge. Each point not fulfilling the last condition is considered as edge points. The third approach is to calculate the change in surface normals in local neighborhoods of a point. If the surface normals differences are significant, the query point is defined as an edge point.

In [113], *Sappa and Devy* segment the point cloud using edges in two main stages. First, they create a binary edge map by defining scan lines, i.e., planar curves, and detecting jump edges within them. A

jump edge is characterized by a discontinuity on the surface using surface normals. The jump edges are approximated as polylines applying quadratic functions building the binary edge map. Next, they outline the edges following a graph strategy. The points of the binary edge map are triangulated over the 2D space. A weighted graph is calculated with the points of the triangular mesh as nodes and links of the mesh as edges. The distance between the two points of the mesh forms the weights of the graph edges. Some optimizations of the graph edges lead point cloud edges of regions.

Wani and Arabnia [156] propose another edge-based approach with an equidepth voxelized point cloud as input. First, two key points are extracted from the voxelized neighborhood with coordinates of the i th voxel as (x_i, y_i) whose

$$\theta_i = \text{acos} \left(\frac{\mathbf{a}_{i,k} \cdot \mathbf{b}_{i,k}}{\|\mathbf{a}_{i,k}\| \cdot \|\mathbf{b}_{i,k}\|} \right), \quad (2.15)$$

with $\mathbf{a}_{i,k} = (x_i - x_{i+k}, y_i - y_{i+k})$ and $\mathbf{b}_{i,k} = (x_i - x_{i-k}, y_i - y_{i-k})$ having k voxels is below a threshold. Then, key points categorized as fold edges, semistep edges, and boundary edges and close key points of the same types are linked. The edges are grown from these key points applying edge masks.

Edge-based methods enable a fast segmentation, but they are not robust when noise or irregular densities occur in point clouds [39, 90]. Noise or irregular densities often occur in on-board LiDAR data.

Region-based Methods: Region-based methods segment the point cloud into different areas starting from one or more points with prominent characteristics, called seed points, and then grow around or subdivide neighboring points with similar characteristics. These characteristics, e.g., surface orientations or curvature values, are related to simple descriptors, refer to Subsection 2.1.1. Region-based methods can be divided into two groups: i) bottom-up methods, and ii) top-down methods. Bottom-up methods start with a few seed points and grow the regions based on similar characteristics of neighboring points. Top-down methods start with assigning every point of the point cloud to one group and subdividing the group into smaller groups.

Besl and Jain [8] introduced the initial bottom-up approach. It comprises two steps. First, points are identified based on their curvature values. Besl and Jain apply differential geometry to the point cloud by computing mean curvatures and Gaussian curvatures with least squares. These are used to extract seed points which define initial regions that are not adjacent and good for surface fitting. Next, they grow the regions based on characteristics of the points starting with the seed points. A variable-order bivariate surface fitting finds the regions of a seed point based on proximity of points and planarity of surfaces. New seed points representing the determined regions the best are detected to repeat the second step until the regions don't change anymore, the surface fitting's order is larger than a threshold, or the maximum number of iterations is reached.

Furthermore, *Tóvári and Pfeifer* [139] introduce another bottom-up approach for airborne LiDAR data. They randomly select seed points and perform the region growing based on three criteria subsequently. Points within a local neighborhood around a seed point are assigned to the seed point's region: i) if its angle difference to the surface normals is smaller than a fixed threshold, ii) if its distance to an adjusting plane, i.e., an orthogonal distance regression plane, is smaller than a fixed threshold, iii) if its distance to the seed point is smaller than a fixed threshold. The region growing is performed until there are no more points fulfilling the criteria.

Vosselman et al. [155] developed a planar surface growing algorithm. In this case, the seed points are selected randomly and then tested if a minimum number of their neighbor points build a plane forming seed surfaces. The region growing is realized based on the proximity of points and perpendicular distance to a fitted plane. This method results in planar segments. *Pu et al.* [98] applied the planar surface growing algorithm of Vosselman et al. to segment terrestrial LiDAR data. They derive five constraint categories to label building features within the segments. i) The size constraint describes the specific size, e.g., of walls, windows, or doors. ii) The position constraint depicts that certain groups

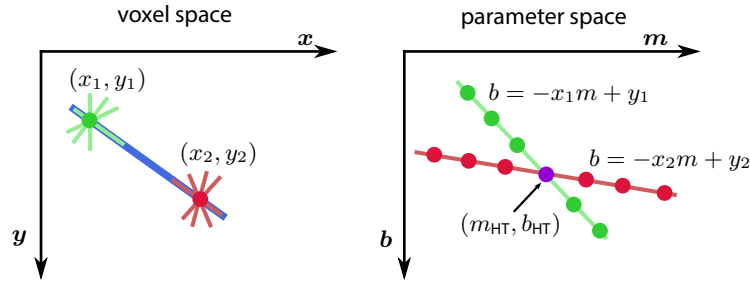


Figure 2.8: 3D HT: All points (green and red) on a mathematical representation (here straight blue line) in the voxel space intersect at a common point (violet) in the parameter space yielding the fitted shape primitive - based on [4].

appear in certain positions, e.g., roofs on top of walls. iii) The direction constraint states that, for example, walls and roofs can be distinguished by their directions, e.g., walls are often vertical. iv) The topology constraint claims that specific segments have topology relations with other segments, e.g., roofs intersect walls. v) The miscellaneous constraint summarizes additional properties, e.g., windows reflect fewer points resulting in a low density compared to solid walls.

Vo *et al.* [153] voxelize the point cloud to incrementally merge neighbor voxels with similar characteristics within the region growing procedure. These characteristics are measured with normal angles applying the PCA and residual values, i.e., the quadratic mean of the perpendicular distances d_i from the query points to the plane fitted into the local neighborhood built by k -nearest neighbors:

$$\sqrt{\frac{1}{k} \sum_{i=1}^k d_i^2}. \quad (2.16)$$

They can be interpreted as simple descriptors, refer to Subsection 2.1.1. The region growing, i.e., merging of voxels with similar characteristics, is realized with an octree approach.

In [7], *Belton and Lichti* combine edge-based segmentation with a bottom-up region-based method. They use points labeled as edges as boundaries for the region growing algorithm.

Chen and Chen [17] introduce an example of a top-down region-based approach. They compute surface normals with a PCA assuming normals of points on a plane point in the same direction. These normals are used to subgroup the point cloud into multiple planar segments.

Both region-based approaches, bottom-up and top-down, require a priori knowledge [39, 90]. Bottom-up approaches need a good start estimation for the seed points, while the main difficulty of the top-down methods is the criteria for the division, e.g., number of regions.

Model-based Methods: Another category of point cloud segmentation algorithms are model-based methods. These methods rely on geometric shape primitives for segmenting the point cloud, e.g., planes, cylinders, and spheres. The segmentation is performed fitting the shape primitives into the point cloud and assigning those points to one segment whose mathematical representation is approximately the same.

A common basis for the fitting of shape primitives are the Hough Transformation (HT) and the RANdom SAmple Consensus (RANSAC) algorithm developed by Ballard [4] and Fischler and Bolles [32], respectively. Both algorithms are used to detect curves like straight lines, circles, or ellipses in local

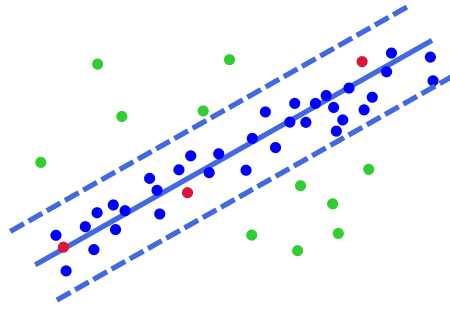


Figure 2.9: RANSAC: First, initial points (red) are selected to fit the mathematical representation of the model (solid blue line) and of the threshold (dashed blue lines). Next, the consensus set (blue points) is assigned to the mathematical representation rejecting outliers (green). The model fitting was successful if the cardinality of the consensus set is large enough - based on [32].

neighborhoods of the point cloud. The HT iteratively tests the mathematical representation of a shape within the voxelized parameter space by counting the best fit for a parameter combination. The parameter combination that occurred most frequently is chosen for the shape primitive Figure 2.8.

The RANSAC fits shape primitives by randomly picking a subset of points building candidates, e.g., least squares. Then, the candidate shapes are checked with all points of the local neighborhood. If the number of points whose distance between those points and the model is smaller than a fixed threshold (consensus set) is larger than a specific amount, the model fitting is successful, refer to Figure 2.9.

Tarsha-Kurdi et al. [130] compare 3D HT and RANSAC algorithms for the detection of roof planes in LiDAR data. In doing so, they discovered that the RANSAC method is more efficient, considering segmentation accuracy and computational effort. The 3D HT method is slower and more sensitive to its parameter values. *Schnabel et al.* [117] propose an approach based on the RANSAC algorithm to detect shape primitives in point clouds as segments. They identify regions that are mathematically represented as planes, spheres, cylinders, cones, tori. In [38], *Gelfand and Guibas* enhance the model representations to slippable shapes. A slippable shape is defined as a rotational and translational symmetrical shape such as spheres, planes, cylinders, and helices. They segment the point cloud by merging initial slippable surfaces to complex shape structures. *Li et al.* [74] based their work on the model fitting algorithm RANSAC as well. The RANSAC is used to fit models into local neighborhoods. Next, a global coupling refines the results of the RANSAC incrementally with correlations between the locally fitted shapes, i.e., orientation, placement, and equality. This method corrects the parameters of the fitted primitives when segmenting point clouds.

Rabbani et al. [102] introduce another model-based approach. Initially, the normal for each point of the point cloud is computed within a local neighborhood with a plane fitting parameterized with the Hesse normal form. The residual in the plane fitting is used to identify highly curved areas. Then, the region growing is performed based on the normal calculation and plane fitting. These characteristics are used to take the local connectivity of areas, and the surface smoothness of areas into account. I. e., the normal differences between the seed point and the points associated with a region, and the residuals are below fixed thresholds, refer to Equation 2.16.

Model-based segmentation is fast and robust with outliers. They are efficient when segmenting point clouds into geometrically simple parameterized shapes, e.g., lines, planes, cylinders, or spheres.

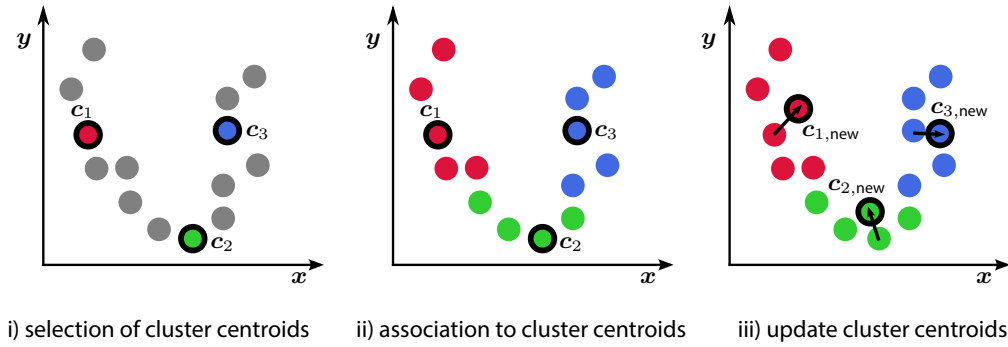


Figure 2.10: The three steps of the k -means algorithm. Here, three clusters are grouped with mean c_i updated at step iii) as $c_{i,new}$ - based on [82].

When dealing with complex shapes, their efficiency and the accuracy of the segmentation results decreases [39, 90]. The application of local descriptors has been proven to provide a more accurate segmentation result for automated implementations and the detection of complex shapes [95].

Clustering Methods: Some segmentation algorithms are based on clustering methods assigned to the unsupervised machine learning category [39]. Their algorithms are designed to form points of several groups, often applying characteristics of the points. Considering point cloud segmentation, hierarchical and especially centroid-based clustering are applied.

The most popular centroid-based clustering method is the k -means algorithm by *Macqueen* [82]. The elements p_i of a data set \mathcal{D} are separated into k partitions, so every element is assigned to one of the k means of the clusters. The k -means algorithm consists of three steps: i) selection of k random cluster centroids $c_{1,\dots,k}$ from \mathcal{D} , ii) associate each element p_i to one of the k clusters by computing the squares of Euclidean distances to the cluster centroids $c_{1,\dots,k,new}$, and iii) update the cluster centroids by calculating the mean of every cluster with the associated points. Step i) and ii) are repeated until the clusters don't change any more. In Figure 2.10, the steps of the k -means algorithm are visualized.

Several works adopted the k -means algorithm for the point cloud segmentation application. *Lavoué et al.* [69] applied the k -means classification to vertex principal curvatures. The algorithm allows them to segment the point cloud by detecting smooth curvature transitions, inflexion points, and regions surrounded by high curvatures. In [19], *Comaniciu and Meer* adjusted the mean shift algorithm to cluster sparse point clouds in a non-parametric way. The mean shift algorithm by *Fukunaga and Hostetler* [36] is an extension of the k -means algorithm. The adoption of *Comaniciu and Meer* applies the mean shift algorithm to search for local maxima of a density function. They want to detect modes in the color or intensity feature space to segment the point cloud. *Yamauchi et al.* [164] rely on the mean shift algorithm as well. For segmenting the point cloud, they cluster mesh normals using the mean shift, thus capturing the point cloud's curvature.

2.1.3. Localization with Non-Semantic LiDAR Data

The following section provides an introduction to the self-localization of automated vehicles relying on non-semantic objects since this thesis focuses on a non-semantic approach based on LiDAR data, called *LiDAR-Feature-based Localization*. A considerable amount of research has been conducted on the self-localization based on non-semantic data. An overview of the studies is presented in this sec-

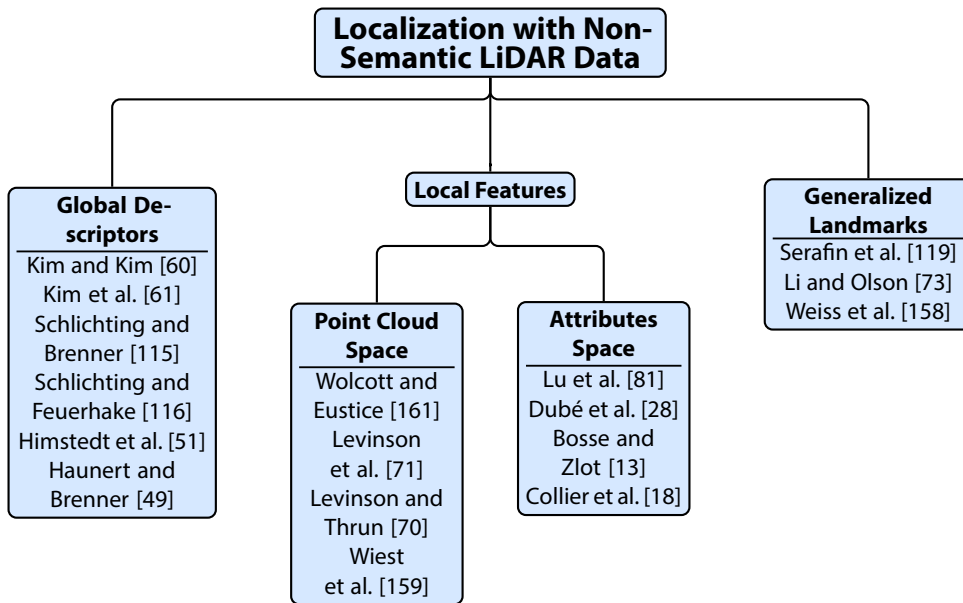


Figure 2.11: Overview of the localization methods based on non-semantic 3D LiDAR data with some examples.

tion. The studies can be divided into three central categories based on the non-semantic data extracted from LiDAR data: i) global descriptors, ii) local features, and iii) generalized landmarks. An overview of the research categories is given in Figure 2.11.

Global Descriptors: Localization solutions based on global descriptors capture characteristics for a whole scene. *Kim and Kim* [60] introduce a global descriptor, called Scan Context, partitioning the LiDAR into sectors with specific azimuth angles and into rings with specific radial values. They call an intersection of a sector and a ring bin, refer to Figure 2.12. Each bin is assigned the largest height of the points falling into that bin forming the description matrix. The vehicle’s localization is realized with a similarity score between Scan Contexts with pairwise scoring and nearest search hierarchically. In further work, *Kim et al.* [61] extend the work on Scan Contexts to Scan Context Images. Here, the Scan Context matrix is normalized and converted into three channels representing the RGB color space. The height is saturated using a jet colormap with a large variance. The localization is carried out using AI, i.e., a Convolutional Neural Network (CNN), learned with LiDAR data recorded during one day.

In [115], *Schlichting and Brenner* detect vertical and planar features via LiDAR sensors. The spatial relations between those vertical and planar features are calculated, representing their scene’s characteristics. The localization is realized, not only matching the local pattern shaped by the extracted features but also associating their descriptions with the map. Another work by *Schlichting and Feuerhake* [116] pursue a non-semantic concept by obtaining descriptions from vertical lines of a 2D LiDAR sensor trained with a neural network (on-board and offline for the map generation). Then, the labeled data is the input for the localization algorithm, in this case, a sequence mining method.

Himstedt et al. [51] follow a similar approach, as in [115]. They extract key points as points with high curvature values within their local neighborhood, i.e., local extrema of one-dimensional range curves. These are the input for the Geometric Landmark Relations (GLARE) estimation, the global description of a scene. The GLARE encodes the spatial relations of the extracted key points. First, the Euclidean

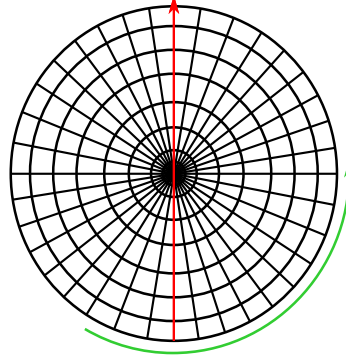


Figure 2.12: In the Scan Context descriptor algorithm, the bins of a scene are divided in radial (red) and azimuth (green) directions - based on [60].

distances $\rho_{i,j}$ of each key points $k_i = \{x_i, y_i\}$ to all other key points $k_j = \{x_j, y_j\}$ for $i \neq j$ are calculated within each scene. Additionally, the bearings $\theta_{i,j}$ and $\theta_{j,i}$ of the corresponding key points are computed

$$\theta_{i,j} = \text{atan2}(y_i - y_j, x_i - x_j), \quad (2.17)$$

while only $\theta_{i,j}^+ = \max(\theta_{i,j}, \theta_{j,i})$ is kept for further calculations. Both the Euclidean distances and the bearings are quantized into uniform bins. Then, the two-dimensional histogram $\mathbf{H}_{i,j}$ at position $m = (m_\rho, m_\theta)$ is generated with a multivariate Gaussian function centered on the bin $n = (n_\rho, n_\theta)$

$$\mathbf{H}_{i,j}(m) = \mathcal{N}(m - n, \mathbf{C}_H) \quad (2.18)$$

with covariance matrix \mathbf{C}_H . The GLARE for a scene is calculated with a normalized sum

$$\eta \sum_i \sum_j \mathbf{H}_{i,j}. \quad (2.19)$$

The GLARE descriptions are collected in a global repository. Scan retrieval with multiple randomized kd-trees yields the vehicles pose.

Hauert and Brenner [49] also pursue a related concept by determining the spatial relation of point landmarks, i.e., pole-like objects like poles of traffic signs or tree trunks. Triangles are used to connect the points creating a point pattern or fingerprint of a scene. The Delaunay triangulation-based fingerprint verification [76] is recreated as the map-based localization solution. Therefore, a set of observed triangles is matched with a set of reference triangles extracted from a pre-built map.

Local Features: Localization solutions relying on local features, extract non-semantic elements with prominent, locally spread characteristics, and use them for the positioning algorithms. There are local features, which are extracted i) based on their characteristics in the attributes space or ii) based on characteristics in the point cloud space. In this work, a distinction between those two types is made in the context of *LiDAR-Feature-based Localization*.

Point Cloud Space: Local features detected in the point cloud space commonly are in the form of grid cells. These localization methods directly quantize the point cloud information. The point cloud data consists of spatial and reflectivity information. For instance, the work of *Wolcott and Eustice* [161] relies on spatial data. They quantize the x and y information of the point cloud into a 2D grid. A grid cell

is assigned a one-dimensional Gaussian mixture function modeling the z , i.e., the height, distribution. In this process, a Gaussian mixture map is generated. The registration of an observation in a Gaussian mixture map establishes the vehicle's pose. The timing efficiency is enhanced with a multiresolution branch-and-bound search.

Moreover, several works are processing the intensity information of the point cloud for the *LiDAR-Feature-based Localization*. *Levinson et al.* [71] quantize the reflectivity data into a 2D x and y grid. In this way, they generate the map using GraphSLAM. The actual localization based on the reflectivity grid is carried out with a particle filter.

In a further development, *Levinson and Thrun* [70] also capture the reflectivity in a 2D x and y grid. Here, in every grid cell not only the average infrared reflectivity of all points falling into that cell is saved but also the variance of the cell's reflectivity. This grid is called a probabilistic map and is generated with the GraphSLAM algorithm. The localization is performed with a 2D histogram filter.

Another localization solution by *Wiest et al.* [159] focuses on reflectivity as input data. In their work, an occupancy grid map based on the reflectivity values is used to detect objects with the Maximally Stable Extremal Regions (MSER) descriptor by *Matas et al.* [83]. Again, a particle filter solves the localization task.

Attribute Space: Local features recognized within the attributes space are based on descriptors or simple versions of them, refer to Subsection 2.1.1. Commonly, simple descriptions such as curvature values are applied here. This is the case in the work of *Bosse and Zlot* [13]. They cluster all regions of the point cloud, which have high curvature values. The curvature is calculated from the second derivative of the LiDAR's range image. *Bosse and Zlot* only keep regions with large positive curvature values as points with large negative curvature often occur due to occlusion boundaries and are not stable. Furthermore, they propose two additional methods of extracting segments from the point cloud. For one thing, they segment the point cloud into regions of connected points whose Euclidean distance is less than 3.5% of their average range. For another, they apply the mean shift algorithm to calculate a locally weighted mean until it converges to a stable point. For each region extracted with each of the three methods, the centroids or the stable points are saved as key points and used in a second step, respectively. The next step is to determine a descriptive characterization given the key point positions. Therefore, *Bosse and Zlot* choose five descriptors: i) Normal Orientation Histogram Grid [80], ii) Orientation and Projection Histograms [12, 157], iii) Hough Transform Peaks [138], iv) gestalt [13], and v) moments grid [13]. These descriptions are calculated on-board and offline as a map. A map voting process is carried out to compute the vehicle pose.

In [18], *Collier et al.* calculate the Variable Dimensional Local Shape Descriptor (VD-LSD) for every point in the LiDAR point cloud. The VD-LSD was developed by *Taati et al.* [129] and is based on the PCA calculations. The covariance matrix is used for the eigenvalue decomposition to calculate an orthonormal frame and the three eigenvalues. This yields to a 22-dimensional description vector, including seven positional, nine rotational, and six dispersion properties. The optimal subset of these characteristics is extracted as in the work of *Taati and Greenspan* [128]. Then, this is used offline as input for the k -means clustering algorithm extracting features for the bag of words algorithm. The bag of words method provides a basis for the ASLAM algorithm built upon the FAB-MAP process, which was initially introduced by *Cummins and Newman* [21]. The bag of words method solves the actual localization solution.

In recent years, the description calculation with neural networks is gaining popularity [100, 101], which also can be used for the non-semantic localization with local features [28, 81]. *Lu et al.* [81] extract local features based on a learned description that is semantically trained in a network. The architecture of their method is visualized in Figure 2.13. They detect a fixed number of key points considering those as key point candidates whose neighborhoods have enough points to build a dense subset of points. From this subset, those points are extracted as key points with strong linear and scattering properties

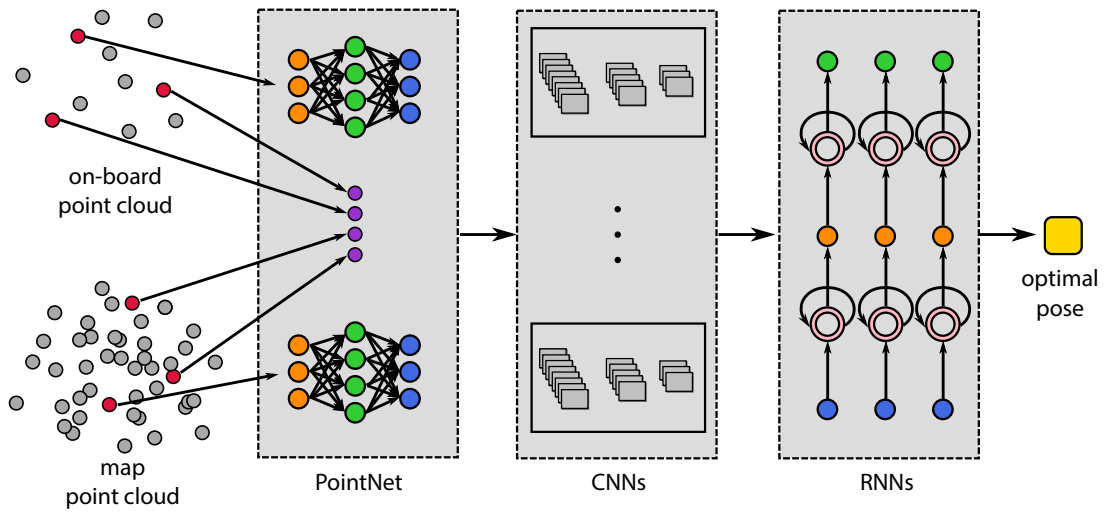


Figure 2.13: The architecture of the AI-based localization with local features by Lu et al.. It takes the on-board, a pre-built map point cloud, and a predicted vehicle pose as inputs to learn descriptions with PointNet and extract feature points (red), construct cost volumes over the solution space, apply CNNs and RNNs to finally estimate the vehicle pose - based on [81].

and are widely scattered in the environment. For each key point, a description is learned based on the PointNet algorithm using semantically labeled, unordered point clouds developed by Qi et al. [100] with a fixed number of 64 neighbors. The localization is also realized with AI with three steps. First, a cost volume and 3D CNNs are constructed to infer and regularize the localization offset configurations for each key point, respectively. Second, a consensus of all key point offsets is found by a probability offset volume for every LiDAR point cloud. The consensus implies the total matching cost between the on-board point cloud and the 3D map specified offset. Third, the sequential LiDAR point clouds are represented as sequential processes with RNNs creating a temporal smoothness.

Dubé et al. [28] as well train a description for a preprocessed point cloud used for localization. Their algorithm, called SegMap, consists of three steps. First, the voxelized point cloud is segmented into regions within a specific radius around the vehicle with an incremental region growing algorithm. In the next step, features are extracted from the point cloud with an AI-based descriptor. Therefore, the point cloud's voxelized segments are transformed into a LRF by applying a PCA. These transformed segments are the input for an Autoencoder encoding the points into the 64-dimensional description space and decoding it to the original voxels' dimensions. Additionally, the inner 64-dimensional layer is the input for classification of the segment realized with batch normalization. Third, candidate correspondences are detected between the offline generated map segments based on dense point clouds and the on-board segments with k -nearest neighbors in the feature space, which are verified with geometric consistency on the segments' centroids. When a set of correspondences is found, a 6 Degree of Freedom (DoF) transformation between the on-board point cloud and the dense map point cloud is executed. This transformation is fed into GraphSLAM, generating a vehicle pose.

Generalized Landmarks: Localization based on generalized landmarks rely on abstractions of landmarks, such as vertical lines, without using semantic class labels. In the work of *Serafin et al.* [119], such localization is introduced. They detect geometric structures based on the assumption of a good feature. That means that the same feature must be identified in most of the point clouds in a set of consec-

utive point clouds while the point clouds might be sparse and noisy. In detail, Serafin et al. establish four criteria defining a good feature: i) sensitivity, ii) repeatability, iii) robustness, and iv) distance measure. Sensitivity means that there are no long periods where no feature is identified. Repeatability means that the same feature is identified under multiple observation states. Robustness means that the same feature is identified under various external conditions like weather or lighting. It should be possible to define a distance measure comparing the characteristics of the feature. Therefore, they identify abstract landmarks as such features, in this case, 3D vertical lines and 3D planes. These generalized landmarks are fed into a GraphSLAM, estimating the vehicle pose.

The generalized landmarks of *Li and Olson* [73] are in the form of edges. First, the point cloud is projected onto the x and y plane and quantized. In the next step, they apply the Kanade-Tomasi detector [133] to detect corners in the 2D plane. They modify the approach to suppress corners that occur based on occlusions.

Weiss et al. [158] don't label the data with semantic information but detect abstract objects, i.e., vertically elongated elements, in a map as well as in on-board sensor data. The map data is used as ground truth, to match the on-board detections with it. The localization consists of five steps. First, the global ego movements, i.e., the odometry in a global coordinate system, are calculated with a pose of a GNSS. Given the global ego movements, the on-board detected generalized landmarks are transformed into the vehicle's local tangent plane. Conclusively, the transformed features are associated with the map features with the Triangle Association algorithm computing and correcting the vehicle's pose.

2.1.4. Conclusions from Relevant Literature for this Thesis

The literature review provides an overview of related work for the idea of a map-based localization relying on non-semantic LiDAR features. In the review, similar works of the concept of *LiDAR-Feature-based Localization* have been identified, the description of LiDAR points, non-semantic extraction of areas of the point cloud, and related localization approaches have been examined. Several conclusions can be drawn from the review, which serve as the basis for the development of a concept for the *LiDAR-Feature-based Localization*. For the following chapters, which present the individual steps of the *LiDAR-Feature-based Localization*, they should be kept in mind. The conclusions are explained hereinafter with the references to the preceding subsections.

Descriptors in Real Data: Many descriptors in the literature are mainly tested on synthetic data unlike the real environment, like sampled Computer-Aided Design (CAD) models. These models create a different testing environment than real data recorded with on-board LiDAR sensors or with realistic synthetic data based on LiDAR models. This discrepancy has three facets.

1. The literature's generated synthetic data compared to real data is not noisy. Besides, in most literature, the sampling is done uniformly, and third, the so generated point clouds are of high resolution. Noise, which is often present in real data, creates scattering within the local neighborhood used for the description calculation. Noise impacts multiple descriptor algorithms capturing the geometric characteristic, e.g., point density or curvature, as they are very sensitive to noise. Thus, the distinctiveness decreases, i.e., a differentiation can only be made between vastly different patterns, not patterns with small differences. For example, in the presence of noise, the descriptions of a planar surface and a slightly curved surface converge.
2. In many cases, the on-board LiDAR sensors' resolution is not uniform resulting in non-equidistant points in the point cloud. Commonly, the horizontal resolution of LiDAR sensors is much higher than the vertical resolution due to the vertically positioned diodes. Non-uniform sampling raises the problem of choosing the local neighborhood's best size, refer to the following conclusion, and induces more sensitivity to the descriptor algorithm. In

particular, this creates effects of the variances of the x , y , and z coordinates resulting in unstable Local Reference Frames (LRFs). It is also a drawback when dealing with quantized local neighborhoods, i.e., voxelized or gridded neighborhoods. In this case, there arise empty or shifted voxels or grid cells which are hard to deal with.

3. The high resolution of the literature's synthetic data captures details that produce a large variance in the description space, which rarely occurs in sparse on-board point clouds, only if the sensor is close to the sampled objects which is seldom the case.

Consequently, the descriptor algorithms' expressiveness, especially their practical relevance for the localization task, needs to be evaluated on real data. Hence, in Subsection 3.2.1 and Subsection 3.3.3, thorough analyzes of multiple descriptors are carried out and discussed regarding their practical relevance for localization on real data. The evaluation includes a viable definition of metrics for the comparison between descriptions and criteria what specifies a good descriptor. Also, several novel descriptors are developed in the course of this thesis, overcoming the shortcomings found in these analyzes.

Local Neighborhood for Descriptors: For the description calculation, most of the examined literature choose either a fixed number of neighbors or a fixed radius for the whole point cloud. When dealing with real data sampled by LiDAR sensors, the environment is often not captured in homogeneous, uniformly distributed point clouds, as the point density depends on the radial distance to the sensors' origin. Points with a large radial distance have a poor sampling density relative to the sensed surface compared to points with a small radial distance. In addition, often, the sensors' resolution is not uniform due to their design. On the one hand, a local neighborhood around each query point of the point cloud built with a fixed number of neighbors does not enclose the same volume throughout the point cloud. A fixed number of neighbors results in differently sized neighborhood sections of the environment. Thus, the description characterizes different parts of the environment, i.e., the description of the same object depends on the object's distance and sensor's sampling rate. Characterizing different parts is not suitable for localization, which should be repeatable under rotations and translations. On the other hand, the same section of the environment is described with a local neighborhood of a fixed size. It is hardly possible to choose a fixed size which is suitable for the whole on-board point cloud. This is the case since the resolution of objects which are far away is much lower than of objects which are close to the sensor. Thus, a fixed neighborhood size for the whole point cloud is limiting when enclosing a sufficiently large local neighborhood while being able to depict details of more densely sampled objects. I. e. a fixed neighborhood forces a weighing of being able to depict details and enabling a sufficient data basis at large distances. This consideration becomes more clear through the example of pillar- and plane-like objects. While the curvature of pillar-like objects with a small radius, e.g., posts of road signs, are hard to describe from far away with few sampling points on it, plane-like objects are still describable from far away.

Concluding, the existing literature does not provide a viable solution on how to choose a local neighborhood size that enables the descriptor to characteristically describe the object. Therefore, Section 3.2 approaches these shortcomings and suggests a solution for a choice of the local neighborhood size.

Features for Localization: Reviewing the literature, features are commonly selected based on their significantly large curvature and surface variation values, as mentioned in the previous conclusion. That means that points lying on planar surfaces and less curved parts are less likely selected. However, points on planar surfaces provide information along their extensions, i.e., orthogonal to their surface normals, which is helpful for the positioning of autonomous vehicles, especially considering the vehicle's heading. In particular, those features should be extracted which are

most suitable and useful for the localization task. Most authors only take the feature's robustness into account when extracting features, see the previous conclusion. The geometric constellation or the identifiability of the features within each scene also influence the localization's performance.

Hence, a definition of the suitability of a feature is a prerequisite for the localization task, while no common definition seems to exist in the related work. Therefore, Section 4.3 defines a metric to extract only those features from the point cloud which are most useful for the localization task.

Robust Extraction of Features: Non-semantic features employed for localization need to be reliably identified under several variations, including viewpoint changes, sensor noise, occlusion, clutter, and point density variation. In the literature, often, features extracted from LiDAR sensors are key points with large curvature and surface variation values. This process is a significant downside for map-based localization since it is difficult to handle a repeatable matching of single key points with map key points using sparse point clouds of LiDAR sensors. This is the case because these single key points need not exist in the sparse point clouds and the closest point may even be far away. Also, commonly, the related work extracts a fixed number of features without considering scene-dependent characteristics. A fixed number may result in extracted features that are unstable. For example, while an urban scene with many anthropogenic objects, such as houses or road signs, contains many robust features, a rural scene only contains a few robust features. Hence, a variable number of features in each scene needs to be detectable. Furthermore, a consistent definition of a robust feature does not exist in the related work, which is a prerequisite for the feature extraction task. While the relevant literature highlights the necessity of a consistent definition, the feasibility and the usefulness vary a lot concerning the application.

In conclusion, the related work does not provide an extraction method for robust features in real-world data in the context of localization. Therefore, Section 4.2 presents an algorithm to detect various feature areas instead of points. It respects characteristics of each scene while applying a definition of the robustness and usefulness of a feature.

Application of Features in Localization: Reviewing the related work, it has been shown that every type of element used for localization, e.g., landmarks, point clouds, or non-semantic features, must be integrated into a convenient localization algorithm. The non-semantic features should be applied to a state-of-the-art localization solution enabling comparability and an evaluation of the research question of this thesis. Especially concerning non-semantic elements, the matching of the on-board detected features with map features is an essential task, which should be performed with high accuracy. In the case of map-based positioning, it is often a problem of keeping the map up-to-date. Therefore, a deterministic localization algorithm with the possibility of updating the map should be employed. Further, since SDSs integrate multiple sensors, an algorithm with an integrated sensor fusion needs to be implemented.

Deriving from the literature review, the *LiDAR-Feature-based Localization* should be realized with a state-of-the-art deterministic localization algorithm enabling a high localization accuracy, fuse multiple sensor data, and a possibility to update the map. These properties are present in the GraphSLAM algorithm, which therefore builds the basis for the self-localization. The integration of the features into the method is presented in Section 5.3. Validation of the *LiDAR-Feature-based Localization* in a real-world data set is outlined in Section 5.2.

Practical Relevance: The existing localization approaches relying on non-semantic LiDAR features have not been tested concerning their practical relevance with regard to existing semantic methods. However, it is necessary to examine the scene coverage of the features compared to infrastructure elements mainly used for the semantic localization to evaluate the contribution of

the non-semantic approach. The examination is especially important since the idea of the non-semantic localization compared to the semantic localization is that it does not limit the range of application to predefined domains due to the semantic knowledge. As the automatic detection of robust and suitable non-semantic features without semantic a priori knowledge is expensive, its benefit needs to be traded off.

Concluding, the practical relevance of non-semantic features for the localization task needs to be evaluated. All of these experiments are performed and demonstrated in Section 5.3.

These conclusions serve as valuable input for the development of the concept of the *LiDAR-Feature-based Localization* and the subsequent analyzes. Chapter 3 takes the existing descriptor algorithms as the starting point to analyze them with realistic synthetic or real data. The applicability of state-of-the-art descriptor algorithms in real-world data can be deduced from these analyzes. They also provide information about possible improvements of state-of-the-art algorithms, like extensions and further developments of the descriptor algorithms. It includes the proposition of a method for choosing a convenient local neighborhood size. In Chapter 4, the identified missing aspects of the feature extraction are integrated into an algorithm to robustly extract features and select those who are most useful for localization. Chapter 5 addresses the application of non-semantic features into an appropriate localization solution factoring in the association of the on-board features to the map and the map updating problems. Section 5.3 outlines the analyzes of the application focusing on the practical relevance of the algorithms.

2.2. Mathematical Fundamentals

In this section, the theoretical basics and algorithms that are employed in the course of this work or contribute to the understanding of this work are introduced.

2.2.1. Coordinate Systems

In this thesis, mainly three reference frames are applied: the Universal Transverse Mercator (UTM), the Sensor Reference Frame (SRF), and the Vehicle Reference Frame (VRF). Figure 2.14 depicts all three coordinate systems.

UTM The Universal Transverse Mercator (UTM) coordinate system is a Cartesian world reference frame $\{x_{UTM}, y_{UTM}, z_{UTM}\}$ describing the vehicle's pose in global coordinates. It divides the earth into 60 vertical zones, each of the zone capturing 6° of longitude in width, and projects each zone to a plane, for the basis of its Cartesian coordinates. Moreover, for each zone, a distinct secant transverse Mercator projection is applied to receive the Cartesian cells, to approximate each zone with the best fitting plane. A sample zone is depicted in Figure 2.14. The x -axis points east, the y -axis points north. Therefore, the x -axis is called easting and the y -axis is called northing. The altitude is captured with the ellipsoid height z_{UTM} . Each of these values are specified in meters. Furthermore, a fourth dimension, measured clockwise from x_{UTM} , serves to describe the vehicle's orientation, also called heading θ_{UTM} . Here, 0° refers to pointing to the east. As the computation of relative distances in an ellipsoid coordinate system, e.g., World Geodetic System of 1984 (WGS84), is a challenging task, the UTM coordinate system is used throughout this thesis. Alternatively, a Local Tangential Plane (LTP) system, also referred to as East North Up (ENU) coordinate system, can be employed.

SRF according to ISO 8855 The coordinate system defined by the ISO 8855 standard [56] is applied to capture the geometric information in a sensor's point of view, mainly in case of the LiDAR sensor. The Cartesian reference frame $\{x_{SRF}, y_{SRF}, z_{SRF}\}$ is illustrated in Figure 2.14. Its origin is centered at the

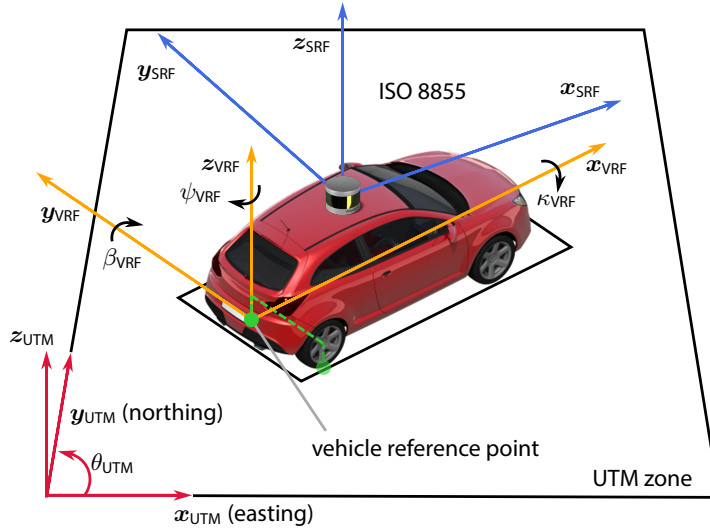


Figure 2.14: The three coordinate systems employed in this thesis: the global UTM system (red), the local SRF (blue), and the local VRF (orange), the latter two according to ISO 8855.

sensor's origin. The coordinate system's orientation defined by the standard is defined by the x_{SRF} -axis, pointing in the specified direction of the sensor. In this thesis, this coordinate system is used to describe the LiDAR point cloud in reference to the sensor.

VRF according to ISO 8855 The coordinate system defined by the ISO 8855 standard [56] is also applied to capture the position in a vehicle's point of view. The Cartesian reference frame $\{x_{\text{VRF}}, y_{\text{VRF}}, z_{\text{VRF}}\}$ is again presented in Figure 2.14. Its origin is centered at the projection of the rear axis' center onto the ground (yellow point in Figure 2.14). The coordinate system's orientation defined by the standard is defined by the x_{VRF} -axis, parallel to the support surface pointing in the direction of the vehicle. The rotation around any of the three axes describes the vehicle's current dynamic state, where κ_{VRF} describes the roll, β_{VRF} the pitch and ψ_{VRF} the yaw of the vehicle. This coordinate system is employed throughout this thesis to depict measurements, like LiDAR point clouds or objects, in reference to the vehicle.

2.2.2. Principal Component Analysis

The Principal Component Analysis (PCA) is the method of determining the principal components in data sets, i.e., the eigenvectors of the covariance matrix of the data, to transform the data into a new basis. The axes of the new reference system are aligned along the principal components according to the existing data structure. In this thesis, the PCA is used to transform an extract of the point cloud with the center of a query point into a viewing-angle independent coordinate system [122]. In the following, the steps of the PCA are explained in more detail, applying it to point cloud data sets as used in this thesis.

Let $\mathcal{P} \subset \mathbb{R}^3$ be a 3D point cloud with m points $\mathbf{p}_1, \dots, \mathbf{p}_m \in \mathbb{R}^3$ with a query point $\mathbf{p}_q \in \mathbb{R}^3$ in the center. In the first step, the query point is shifted to the origin of the point cloud by subtracting it from each point \mathbf{p}_i :

$$\mathbf{p}'_i = \mathbf{p}_i - \mathbf{p}_q \quad \forall i = 1, \dots, m. \quad (2.20)$$

The covariance matrix $\mathbf{C} \in \mathbb{R}^{3 \times 3}$ is determined of the shifted points \mathbf{p}'_i .

By solving the characteristic equation, the eigenvalues λ can be calculated from the covariance matrix C :

$$\det(\lambda \mathbb{1} - C) = 0. \quad (2.21)$$

The eigenvectors e_i result as solutions of the linear equation

$$(\lambda_i \mathbb{1} - C)e_i = 0. \quad (2.22)$$

The eigenvectors are normalized to length $|e_i| = 1$. They are chosen so that the following applies:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3. \quad (2.23)$$

The eigenvector corresponding to the greatest eigenvalue represents the the first principal component of the point cloud. This is the direction with the greatest variance within the data. As the eigenvalue decreases, the variance decreases in the direction of the associated eigenvector, so that the eigenvector with the smallest eigenvalue corresponds to the direction with the lowest variance.

The eigenvectors, i.e., principal components, are used perform a change of basis into the new reference system on the data by multiplying $E = [e_1 \ e_2 \ e_3]$ with the initially shifted points p'_i and resifting the point cloud to its initial origin:

$$p_{i\text{trans}} = (E^T \cdot p'_i)^T + p_q, \forall i = 1, \dots, m. \quad (2.24)$$

2.2.3. Iterative Closest Point

A fundamental concept when dealing with LiDAR point clouds is the Iterative Closest Point (ICP) algorithm registering 3D point clouds developed by Besl and McKay [9], which is explained in this section.

The ICP aims to estimate the allocation of two point clouds by determining the translation vector and the rotation matrix for the transformation from one point cloud to the other given two corresponding point clouds, the target point cloud $C_1 = \{p_{1,1}, \dots, p_{1,n}\}$ and the reference point cloud $C_2 = \{p_{2,1}, \dots, p_{2,n}\}$.

To begin with, the target point cloud is initialized for the first iteration of the ICP algorithm:

0. The target point cloud is initialized with

$$C_1^0 = \{p_{1,1}^0, \dots, p_{1,n}^0\} = \{p_{1,1}, \dots, p_{1,n}\}. \quad (2.25)$$

Then the ICP algorithm is carried out in four steps for every iteration step k :

1. For each point of C_2 the closest point in C_1^k is determined, i.e., minimizing $\|p_{1,j}^k - p_{2,i}\|_2$ for all i, j , allowing ambiguous assignments.
2. The rotation matrix R^k and the translation vector t^k is calculated minimizing the sum of the squared error

$$e_k = \frac{1}{n} \sum_{i=1}^n \|R^k p_{1,i}^k + t^k - p_{2,i}\|^2. \quad (2.26)$$

3. The point cloud C_1^k is shifted and rotated corresponding to the transformation parameters of the previous step

$$C_1^{k+1} = \{p_{1,i}^{k+1} \mid p_{1,i}^{k+1} = R^k p_{1,i}^k + t^k, p_{1,i}^k \in C_1^k\}. \quad (2.27)$$

4. The distance d^{k+1} between the two point clouds is calculated

$$d^{k+1} = \sum_{i=1}^n \|\mathbf{p}_{1,i}^{k+1} - \mathbf{p}_{2,i}\|^2. \quad (2.28)$$

The steps of the ICP are repeated, until d^{k+1} is smaller than a given threshold ε or k is higher than the preset maximum number of iterations. Optionally, instead of considering the distance between every point of the two point clouds, just the point pairs with $100 - \alpha\%$ of the smallest distances can be taken into account with a rejection parameter of $\alpha\%$.

A sufficient initial registration of both point clouds is necessary for an accurate ICP registration [9]. I. e. too large offsets or different orientations of the point cloud lead to imprecise registration results. Additionally, the ICP algorithm assumes that all points of the target point cloud correspond to the reference point cloud.

Overcoming these issues, a **modified version of the ICP** by Wilbers et al. [160] is applied in this thesis. Here, the point cloud is rotated with every rotation matrix \mathbf{R}^k for a specific number of angles in a preset interval. The algorithm is carried out in three steps for every iteration step k to find the best shift concerning each rotation:

1. The Euclidean distances e_j^k between every point of the rotated target point cloud \mathcal{C}'_1^{k+1} and every point of the reference point cloud \mathcal{C}_2 , i.e., $\forall j = 1, \dots, n$, are calculated

$$\begin{aligned} \mathcal{C}'_1^{k+1} &= \left\{ \mathbf{p}'_{1,i}{}^{k+1} \mid \mathbf{p}'_{1,i}{}^{k+1} = \mathbf{R}^k \mathbf{p}_{1,i}^k, \mathbf{p}_{1,i}^k \in \mathcal{C}_1^k \right\}, \\ \mathbf{e}_j^k &= \left[\|\mathbf{p}'_{1,1}{}^{k+1} - \mathbf{p}_{2,j}\|_2 \quad \dots \quad \|\mathbf{p}'_{1,n}{}^{k+1} - \mathbf{p}_{2,j}\|_2 \right]^\top. \end{aligned} \quad (2.29)$$

2. Every point of the processed target point cloud \mathcal{C}'_1^{k+1} is shifted with each Euclidean distance vector \mathbf{e}_j^k , i.e., $\forall j = 1, \dots, n$ for each vector whose norm is smaller than a preset range threshold

$$\mathcal{C}_j^{k+1} = \left\{ \mathbf{p}_{1,i}^{k+1} \mid \mathbf{p}_{1,i}^{k+1} = \mathbf{p}'_{1,i}{}^{k+1} + \mathbf{e}_j^k, \mathbf{p}'_{1,i}{}^{k+1} \in \mathcal{C}'_1^{k+1} \right\}. \quad (2.30)$$

3. The best shift is calculated minimizing the mean distances d_j^{k+1} between the points of each shifted point cloud \mathcal{C}_j^{k+1} , i.e., $\forall j = 1, \dots, n$, and the reference point cloud \mathcal{C}_2

$$\begin{aligned} d_j^{k+1} &= \frac{1}{n} \sum_{i=1}^n \|\mathbf{p}_{1,i}^{k+1} - \mathbf{p}_{2,j}\|_2^2 \\ \mathbf{e}_{\min}^k &= \underset{\mathbf{e}_j^k}{\operatorname{argmin}} d_j^{k+1}. \end{aligned} \quad (2.31)$$

These steps are performed for every angle of the rotation matrix of a predefined interval. The point cloud rotated with the angle and shifted with the translation vector belonging to the smallest mean distance of Equation 2.31 over all iteration steps represents the registered point cloud. Optional, if a distance of $\|\mathbf{p}_{1,i}^{k+1} - \mathbf{p}_{2,j}\|_2^2$ is greater than a preset threshold, a penalty factor can be multiplied with the distance. In this way, a possible non-match between the point clouds, specified with the preset threshold, is punished severely.

Additionally, this procedure can be extended to work with additional elements besides points, like lines. In this case, the points are still used to find the best shift, but the additional elements are consid-

Table 2.1: Parameters of the Daisy description calculation - based on [132].

parameter	symbol	explanation
radius	r	Distance from query pixel to outermost grid point
radius quantization	q	Number of standard deviations for smoothing of orientation maps (layers)
angle quantization	t	Number of considered pixel per layer
histogram quantization	h	Number of gradient orientations and thus of orientation maps
number of grid points	s	Total number of considered pixels depends on q and t
size of description	d	Total size of description depends on s and h

ered when calculating the mean distance d_j^{k+1} in step 3, see Equation 2.31. For instance, an average can be built of the Euclidean distance between points and the perpendicular distance between lines.

This algorithm finds the global solution within the predefined rotation interval but has a high computational effort.

2.2.4. Ramer-Douglas-Peucker Algorithm

The Ramer-Douglas-Peucker algorithm by Douglas and Peucker, Ramer is a calculation specification designed to decrease the number of points of a curve to an approximated curve with fewer points [27, 103].

The algorithm recursively partitions the curve. Initially, the curve $c \in \mathbb{R}^{2+}$ consists of an ordered set of n points $p_1, \dots, p_n \in \mathbb{R}^2$: $c = \{p_1, \dots, p_n\}$ and a threshold $\varepsilon > 0$ is defined. As an approximation of c , the distance from the first to the last point $\overline{p_1 p_n}$ is considered. Both points are marked as to be kept for the approximation. In order to check whether this approximation is sufficient, the point of the remaining subset of inner points $c \setminus \{p_1, p_n\}$, i.e., the points that are not marked as to be kept, is determined which is farthest from the line segment. Here, different distance metrics can be used, e.g., the perpendicular distance or the shortest distance from a point to a line segment. If this point is closer to the line segment than ε , all points, which are not marked as to be kept, can be dismissed. Otherwise, the point is marked as to be kept and the algorithm recursively carries out the same steps for the first point and the farthest point and subsequent with the farthest point and the last point. After the recursion is finished the output curve is defined as the ordered set of all points which are marked as to be kept.

During this thesis, the Ramer-Douglas-Peucker algorithm is used to abstract the shape of multiple points of a point cloud in 2D to a curve.

2.2.5. Daisy Descriptor

A popular descriptor for image processing, i.e., processing two-dimensional, rectangular pixel grids, the Daisy descriptor by Tola et al. [132], is presented here. Its computation is based on the greyscale gradients of the image pixels using differently parameterized Gaussian smoothing kernels. Starting with the gradient of the query pixel, further gradients of surrounding pixels are considered. The algorithm is carried out in three steps.

First, the local neighborhood of the query point is built. Therefore, the number and positions of the surrounding image values, whose greyscale gradients should be considered besides the query pixel's gradients, are determined. As the positions are formed using a radial grid, see Figure 2.15, the considered neighbor pixels are also called grid points. The number and positions depend on preset parameters, the maximal distance to the position r , the quantization parameter for the radius q and for the angle t , which are outlined in Table 2.1. The total number of grid points s is determined from

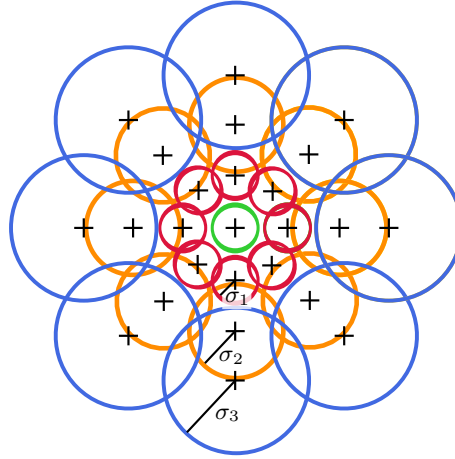


Figure 2.15: Structure of the considered pixels (crosses) with three layers or standard deviations $q = 3$ (each in the same color) and eight orientations $t = 8$ used for the Daisy description calculation - based on [132].

the number of radial partitions q , called layers, and the number of azimuth partitions t :

$$s = q \cdot t + 1, \quad (2.32)$$

with regard to the query point. The coordinates $(x_{i,\text{grid}}, y_{i,\text{grid}})$ of each grid point with $i = 1, \dots, t$ are calculated with the following formula:

$$\begin{aligned} x_{i,\text{grid}} &= \frac{r}{q} \cdot \cos\left(2 \cdot i \cdot \frac{\pi}{t}\right) \\ y_{i,\text{grid}} &= \frac{r}{q} \cdot \sin\left(2 \cdot i \cdot \frac{\pi}{t}\right). \end{aligned} \quad (2.33)$$

The positions of all grid points also define the size of the differently parameterized Gaussian smoothing kernels, see circles of Figure 2.15. The further away a grid point is, the larger the standard deviation of the Gaussian kernel gets. In Figure 2.15, the positions of the grid points and the standard deviations are visualized. They form the shape of a daisy, which is where the name of the descriptor originates.

Second, gradient images of the input images are calculated and smoothed with different Gaussian kernels. Therefore, the input image is smoothed with standard deviation $\sigma = 0.5$. The smoothing of images is performed by a convolution with a 2D Gaussian kernel with a specified standard deviation. These gradient images G_i in h evenly distributed orientations are calculated based on the smoothed input image I :

$$G_i = \left(\cos(\theta_i) \frac{\delta I}{\delta x} + \sin(\theta_i) \frac{\delta I}{\delta y} \right)^+, \quad (2.34)$$

for every angle $\theta_{1,\dots,h}$ with $(a)^+ = \max(a, 0)$. They are called orientation maps. The numerical differentiation is performed with a convolution of I with the h -times rotated vector $[0.5, 0, 0.5]$. The gradient images are then smoothed with a standard deviation of $\sigma \approx 1.5$, resulting in a total standard deviation of $\sigma_{\text{tot}} = \sqrt{0.5^2 + 1.5^2} \approx 1.6$ regarding the smoothing of the input image. Next, these smoothed

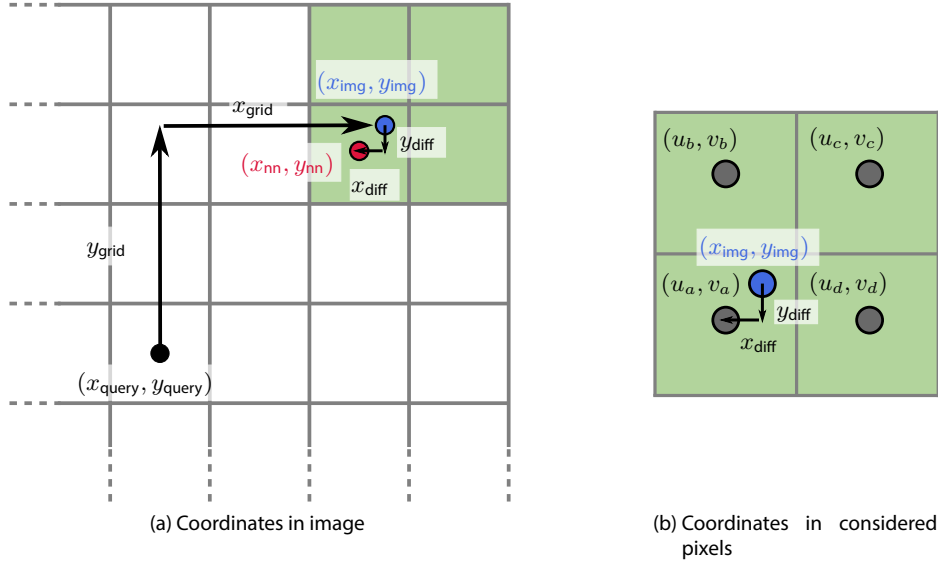


Figure 2.16: Illustration of the coordinates of the considered pixels (green) for the calculation of the Daisy descriptor - based on [132].

images are again smoothed with q different standard deviations σ_i :

$$\sigma_i = \frac{r(i+1)}{2q} \quad \forall i = 1, \dots, q. \quad (2.35)$$

The standard deviations are proportional to the size of the considered neighborhood r , and there exist q standard deviations corresponding to the radial quantization. These multiple convolutions with the Gaussian kernel are divided into consecutive convolutions to reduce the computational effort. Thus, the standard deviations σ_j of the consecutive convolutions need to be determined:

$$\sigma_j = \sqrt{\sigma_2^2 - \sigma_1^2}, \quad \text{with } \sigma_2 > \sigma_1. \quad (2.36)$$

The size of each convolution kernel k_i depends on the determined standard deviation:

$$k_i = \begin{cases} \max(\lfloor 5\sigma_i \rfloor, 3) + 1, & \text{if } \text{mod}(k, 2) = 0 \\ \max(\lfloor 5\sigma_i \rfloor, 3), & \text{otherwise.} \end{cases} \quad (2.37)$$

Consequently, each of the h orientation maps, i.e., gradient images, exist with q different smoothing applications.

Third, the actual description of size d is determined based on the gradient images by examining gradient values neighboring coordinates within the grid cells starting with the query pixel. Therefore, the coordinate within the grid cells $(x_{i,\text{img}}, y_{i,\text{img}})$ is computed for every offset from the query pixel to the considered pixel $(x_{i,\text{grid}}, y_{i,\text{grid}})$ with $i = 1, \dots, t$ (cf. Figure 2.16a):

$$\begin{aligned} x_{i,\text{img}} &= x_{i,\text{query}} + x_{i,\text{grid}} \\ y_{i,\text{img}} &= y_{i,\text{query}} + x_{i,\text{grid}}, \end{aligned} \quad (2.38)$$

with $(x_{i,\text{query}}, y_{i,\text{query}})$ representing the coordinates of the query pixel. See 2.16a for the visualization of the coordinates within the grid. As the determined coordinates $(x_{i,\text{img}}, y_{i,\text{img}})$ do not fall within the center of a pixel, the differences $(x_{i,\text{diff}}, y_{i,\text{diff}})$ between the determined coordinates $(x_{i,\text{img}}, y_{i,\text{img}})$ and the pixel around the determined pixel with coordinates $(x_{i,\text{nn}}, y_{i,\text{nn}})$, which is nearest to the center pixel, are computed (cf. 2.16a):

$$\begin{aligned} x_{i,\text{nn}} &= \lfloor x_{i,\text{img}} \rfloor, \\ y_{i,\text{nn}} &= \lfloor y_{i,\text{img}} \rfloor, \\ x_{i,\text{diff}} &= x_{i,\text{img}} - x_{i,\text{nn}}, \\ y_{i,\text{diff}} &= y_{i,\text{img}} - y_{i,\text{nn}}. \end{aligned} \quad (2.39)$$

Therefore, the gradient values of the pixels surrounding the determined positions, as in Equation 2.38, are considered. The column vectors $\mathbf{h}_{i,a}$, $\mathbf{h}_{i,b}$, $\mathbf{h}_{i,c}$, and $\mathbf{h}_{i,d}$ specify the gradients of the pixels $(u_{i,a}, v_{i,a})$, $(u_{i,b}, v_{i,b})$, $(u_{i,c}, v_{i,c})$, and $(u_{i,d}, v_{i,d})$ for h orientations. With the weights of the distances $x_{i,\text{diff}}$ and $y_{i,\text{diff}}$ the gradient vector \mathbf{h}_i is calculated. This gradient vector represents the vector for the originally determined coordinates $(x_{i,\text{img}}, y_{i,\text{img}})$ (cf. 2.16b):

$$\mathbf{h}_i = (1 - y_{i,\text{diff}}) \cdot (x_{i,\text{diff}} \cdot \mathbf{h}_{i,c} + (1 - x_{i,\text{diff}}) \cdot \mathbf{h}_{i,a}) + y_{i,\text{diff}} \cdot (x_{i,\text{diff}} \cdot \mathbf{h}_{i,d} + (1 - x_{i,\text{diff}}) \cdot \mathbf{h}_{i,b}). \quad (2.40)$$

The resulting vector \mathbf{h}_i of length h builds one row of the final description, but the gradients are based on the differently smoothed gradient images (cf. Equation 2.35). The calculations of Equation 2.40 are carried out for each of the $s = q \cdot t + 1$ grid points resulting in s description rows and thus a description matrix of size $s \times h$. This yields, refer to Figure 2.15, to the Daisy descriptor matrix \mathbf{D} for the query point of:

$$\mathbf{D} = \left(\hat{\mathbf{h}}_{\sigma_1}^T \quad \hat{\mathbf{h}}_{\sigma_1}^T(l_1(r_1)) \quad \dots \quad \hat{\mathbf{h}}_{\sigma_1}^T(l_t(r_1)) \quad \dots \quad \hat{\mathbf{h}}_{\sigma_q}^T(l_1(r_q)) \quad \dots \quad \hat{\mathbf{h}}_{\sigma_q}^T(l_t(r_q)) \right)^T, \quad (2.41)$$

with q as the number of circular layers, r_i the corresponding radii, and σ_i the corresponding standard deviation for the Gaussian kernels, as in Figure 2.15. The position on the layer is specified by $l_j(r_i)$ with $j = 1, \dots, t$ as angle quantization number. Every row of the description $\hat{\mathbf{h}}_{\sigma_1}^T$ is normalized with the vectors norm for every pixel to reduce occlusion effects, which is denoted with $(\hat{\cdot})$.

2.2.6. Density-based Spatial Clustering of Applications with Noise

For the extraction of features, point cloud clustering methods are often applied. This is also the case in this thesis. Therefore, first, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm by Ester et al. [29] is outlined hereafter.

The DBSCAN is a density-based clustering approach grouping points together which are close to each other while detecting outliers. This approach underlies the assumption that the point density of a cluster is greater than outside the cluster. Ester et al. use the number of points within the neighborhood with the preset radius ε to define the density of a point \mathbf{p}_i , called ε -environment. At least a preset threshold m points have to be within the neighborhood to build a cluster.

Ester et al. define three types of point categories: i) core point, ii) border point, iii) outlier. A core point is defined as a point whose number of points within the ε -environment amounts at least m . A border point is no core point but lies within the ε -environment of some core point. An outlier is neither a core point nor a border point.

The idea of the DBSCAN is that two core points whose distance is at most ε are assigned the same cluster, border points are assigned the cluster of the correspondent core point, and outliers are neglected. The algorithm consists of five steps:

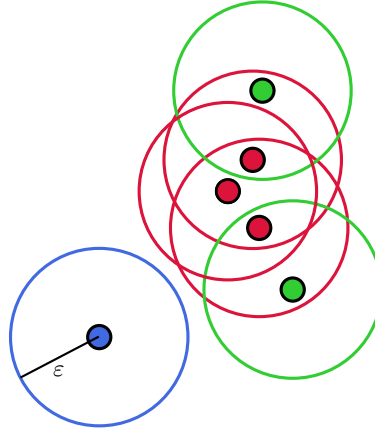


Figure 2.17: The three point categories used in the DBSCAN algorithm with $m = 3$ including core points (red), border points (green), and noise (blue) with their ε -environment (circles) - based on [29].

1. The data points are identified as core points, border points, or outliers.
2. All outliers are subtracted from the data set.
3. An edge connects all core points lying within an ε -environment.
4. A set of connected core points build a separate cluster.
5. The border points are assigned the cluster with the adjacent core point.

The DBSCAN successfully filters noise and can group clusters of any form and size, but fails in the case of clusters with different densities.

2.2.7. k-Medoids

Another clustering method by Kaufman and Rousseeuw [59], called k -medoids, is described in this subsection. The k -medoids algorithm partitions a data set into k clusters. Each cluster is represented by one instance, i.e., data point, of the cluster. This data point has a minimal average distance to all other data points within the cluster. It is called a medoid. The algorithm is performed in three steps, which are visualized in Figure 2.18:

1. k cluster representatives, i.e., medoids, $\mathcal{M} = \{\mathbf{m}_1, \dots, \mathbf{m}_k\}$ are selected randomly from the data set $\mathcal{C} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$.
2. Assign every data point to the nearest medoid using a distance metric. This yields the subsets of data points associated to \mathbf{m}_i with $i = 1, \dots, k$, i.e., clusters \mathcal{C}_i .
3. Let $\text{dist}(\cdot, \cdot)$ be a distance function between two data points. A medoid \mathbf{m}_i is replaced by another point $\mathbf{p} \in \mathcal{C}_i$ of cluster \mathcal{C}_i if

$$\sum_{\mathbf{p}' \in \mathcal{C}_i} \text{dist}(\mathbf{p}', \mathbf{p}) < \sum_{\mathbf{p}' \in \mathcal{C}_i} \text{dist}(\mathbf{p}', \mathbf{m}_i) \quad \forall \mathcal{C}_i \quad \forall \mathbf{p} \in \mathcal{C}_i \setminus \{\mathbf{m}_i\}. \quad (2.42)$$

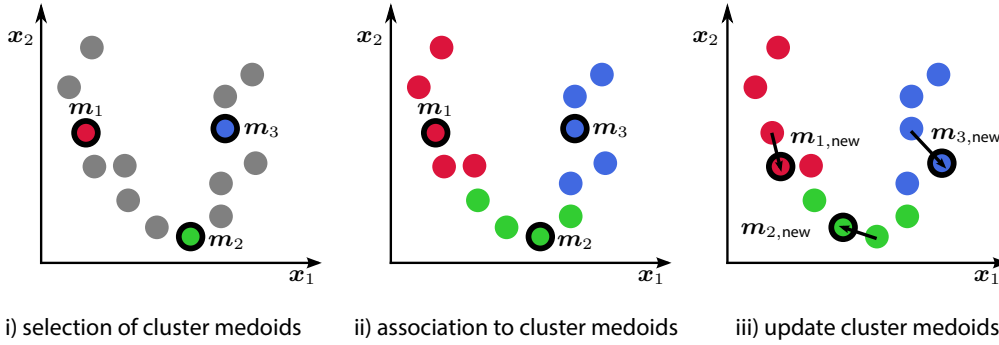


Figure 2.18: The three steps of the k -medoid algorithm. Here, three clusters are grouped with medoid m_i updated at step iii) as $m_{i,new}$ - based on [59].

The result of this original algorithm depends heavily on the initial selection of the medoids. Therefore, some improvements have been made [59, 89].

2.2.8. Graph-based Simultaneous Localization and Mapping

In this subsection, the details of the Graph-based Simultaneous Localization And Mapping (SLAM) are presented [40, 131]. The problem of SLAM is to solve the positioning of a robot or vehicle and mapping at the same time. Measurements of different on-board sensors are the only information of the environment, which are afflicted with errors and inaccuracies. The handling of these is the main issue of SLAM [131].

In the literature, a differentiation is made between two definitions of the SLAM problem [40, 131]. The first definition is called *full SLAM*. Solving the full SLAM problem includes estimating the probability of the robot's total trajectory and the map given all measurements and the initial position. The second definition is called *online SLAM*. Here, the problem to be solved consists of the estimation of the robot's pose at one timestamp given the current measurements and the map. In addition to the map, only the current pose and the associated measurements are relevant here. In recent years, multiple methods were developed to solve the SLAM problem. The best-known methods are the Extended Kalman Filter (EKF), the particle filter, and the graph-based approach [131]. The filter-based approaches only solve the online SLAM problem, while the smoothing-based approach enables the full SLAM solution. Therefore, in this thesis, the smoothing-based approach is applied. The intuitive way to formulate the smoothing-based approach is the graph-based method, which is employed in this thesis and presented in this section.

Problem Formulation: The problem of SLAM is to estimate the state vector of a robot $x = (x_1, \dots, x_m)^T$, i.e., the robot's pose and its on-board observations, given their measurements z [40]. Let z_i be a measurement of the i -th state x_i and $h_i(x)$ be a (nonlinear) function that predicts the measurement depending on the state vector x . The problem is formulated as a Maximum A Posteriori (MAP) approach finding the most probable state x^* that maximizes the probability under the condition of all measurement states z , i.e., which explains the measurements the best:

$$x^* = \underset{x}{\operatorname{argmax}} p(x|z), \quad (2.43)$$

where $p(x|z)$ is called posterior distribution.

The state vector \mathbf{x}^* is specified as the state with the smallest global error. Therefore, the error function is defined as the sum of the squared error terms of the individual factors $e_i(\mathbf{x}, \mathbf{z}_i)$, which are defined as the difference between the actual and the predicted measurements:

$$e(h_i(\mathbf{x}), \mathbf{z}_i) = h_i(\mathbf{x}) - \mathbf{z}_i. \quad (2.44)$$

In case of additional information like probability distributions of measurement uncertainties, the individual error terms of the squared error terms can be weighted with the information matrix $\mathbf{\Omega}_i$ of the i -th measurement:

$$e_i(\mathbf{x}, \mathbf{z}_i) = e(h_i(\mathbf{x}), \mathbf{z}_i)^\top \mathbf{\Omega}_i e(h_i(\mathbf{x}), \mathbf{z}_i). \quad (2.45)$$

Often, $\mathbf{\Omega}_i$ is chosen as the inverse of the measurement's covariance matrix \mathbf{C}_i .

Deriving the numerical problem: In the following, the conversion of the MAP approach to the minimization of the weighted squared error terms is depicted. The derivation is based on the work of the working group at Volkswagen and Grisetti et al., Thrun and Leonard [40, 131]. Using Bayes rule, the problem Equation 2.43 is transformed into

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{z}|\mathbf{x})p(\mathbf{x}), \quad (2.46)$$

where $p(\mathbf{z}|\mathbf{x})$ is named likelihood distribution and $p(\mathbf{x})$ is named prior distribution. Three assumptions about these distributions are made [40]:

Assumption 2.1 (Independent Measurements and Terms). *The measurement and prior terms are independent and identically distributed.*

As a result, the individual probabilities can be multiplied to form the overall probability:

$$\begin{aligned} p(\mathbf{x}) &= \prod_j p(\mathbf{x}_j) \\ p(\mathbf{z}|\mathbf{x}) &= \prod_i p(\mathbf{z}_i|\mathbf{x}). \end{aligned} \quad (2.47)$$

Assumption 2.2 (Gaussian Distributions). *The likelihood distributions are Gaussian.*

The likelihood distributions are defined as:

$$\begin{aligned} p(\mathbf{z}_i|\mathbf{x}) &= \mathcal{N}(\mathbf{z}_i|h_i(\mathbf{x}), \mathbf{C}_i) \sim \exp\left(-\frac{1}{2}(h_i(\mathbf{x}) - \mathbf{z}_i)^\top \mathbf{C}_i^{-1}(h_i(\mathbf{x}) - \mathbf{z}_i)\right) \\ &= \exp\left(-\frac{1}{2}e(h_i(\mathbf{x}), \mathbf{z}_i)^\top \mathbf{\Omega}_i e(h_i(\mathbf{x}), \mathbf{z}_i)\right). \end{aligned} \quad (2.48)$$

Assumption 2.3 (Gaussian or Uniform Distributions). *The prior distributions are either Gaussian or uniform.*

The prior distributions are defined as either:

$$p(\mathbf{x}_j) = \mathcal{N}(\mathbf{x}_j|\boldsymbol{\mu}_j, \mathbf{C}_j) \sim \exp\left(-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_j)^\top \mathbf{C}_j^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_j)\right) \quad (2.49)$$

or

$$p(\mathbf{x}_j) = \varepsilon, \quad (2.50)$$

with error function $e(\mathbf{x}_j)$ with consistent notation according to Equation 2.44.

The option between a uniform and a Gaussian prior for each considered state enables to apply prior knowledge in the practical usage whenever it exists. For the following derivation, Gaussian prior distributions are assumed. Otherwise, if uniform distributions are assumed, the according terms cancel out such that the following derivation still is correct. Under the assumptions, Equation 2.43 is transformed into:

$$\begin{aligned}
\mathbf{x}^* &= \operatorname{argmax}_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}) \\
&\stackrel{(2.46)}{=} \operatorname{argmax}_{\mathbf{x}} (p(\mathbf{z}|\mathbf{x})p(\mathbf{x})) \\
&\stackrel{(2.47)}{=} \operatorname{argmax}_{\mathbf{x}} \left(\prod_i p(\mathbf{z}_i|\mathbf{x}) \prod_j p(\mathbf{x}_j) \right) \\
&= \operatorname{argmin}_{\mathbf{x}} \left(-\log \left(\prod_i p(\mathbf{z}_i|\mathbf{x}) \prod_j p(\mathbf{x}_j) \right) \right) \tag{2.51} \\
&= \operatorname{argmin}_{\mathbf{x}} \left(-\left(\sum_i \log(p(\mathbf{z}_i|\mathbf{x})) + \sum_j \log(p(\mathbf{x}_j)) \right) \right) \\
&= \operatorname{argmin}_{\mathbf{x}} \left(\sum_i \mathbf{e}(h_i(\mathbf{x}), \mathbf{z}_i)^\top \boldsymbol{\Omega}_i \mathbf{e}(h_i(\mathbf{x}), \mathbf{z}_i) + \sum_j \mathbf{e}(\mathbf{x}_j)^\top \boldsymbol{\Omega}_j \mathbf{e}(\mathbf{x}_j) \right).
\end{aligned}$$

Representing the error vector \mathbf{e}_k with $k = \{i, j\}$ as the results of either $\mathbf{e}(h_i(\mathbf{x}), \mathbf{z}_i)$ or $\mathbf{e}(\mathbf{x}_j)$ the MAP approach of Equation 2.43 can be written as:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \sum_k \mathbf{e}_k^\top \boldsymbol{\Omega}_k \mathbf{e}_k. \tag{2.52}$$

Often, the error vector is called global error function $F(\mathbf{x})$:

$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} F(\mathbf{x}). \tag{2.53}$$

Concluding, it can be shown, that under the Assumptions 2.1, 2.2, and 2.3 finding the most probable state \mathbf{x}^* is equivalent to finding the state \mathbf{x}^* which minimizes the sum of the weighted squared error terms.

Problem Solution: For the solution of the SLAM problem of Equation 2.53, the global minimum of the sum of the weighted squared error terms must be found. As there is often no closed-form solution to this non-linear minimization problem, in the following a numerical iterative method, the Gauss-Newton method, is presented.

The Gauss-Newton method approximates the solution of Equation 2.53 in an iterative process. It consists of five steps.

First, the initial state $\check{\mathbf{x}}$ is estimated. In the very first iteration, for instance, this could be a GNSS pose or a preceding pose predicted to the current time with odometry data.

The second step deals with the linearization of the error terms as the non-linear components of the global error function are the individual error terms. This is realized by the approximation to a first-order Taylor series expansion about $\check{\mathbf{x}}$: $F(\check{\mathbf{x}} + \Delta\mathbf{x}) \approx F_{\text{lin}}(\check{\mathbf{x}}, \Delta\mathbf{x})$. It is performed by analyzing the global

error function at the initial state \check{x} with an added update vector Δx :

$$F(\check{x} + \Delta x) = \sum_i e(\check{x} + \Delta x, z_i)^\top \Omega_i e(\check{x} + \Delta x, z_i) \quad (2.54)$$

The linearization of $e(\check{x} + \Delta x, z_i)$ with the Taylor series expansion yields:

$$e(\check{x} + \Delta x, z_i) \approx e(\check{x}) + \check{J}_i \Delta x, \quad (2.55)$$

with Jacobian matrix $\check{J}_i = \left. \frac{\partial e(x, z_i)}{\partial x} \right|_{x=\check{x}}$ assuming the update vector is small. Substituting Equation 2.55 in Equation 2.54 results in the linearized global error function $F_{\text{lin}}(\Delta x)$:

$$\begin{aligned} F(\check{x} + \Delta x) &\approx F_{\text{lin}}(\Delta x) = \sum_i (e(\check{x}) + \check{J}_i \Delta x)^\top \Omega_i (e(\check{x}) + \check{J}_i \Delta x) \\ &= \sum_i \underbrace{e(\check{x})^\top \Omega_i e(\check{x})}_{=: c_i} + 2 \underbrace{e(\check{x})^\top \Omega_i \check{J}_i}_{=: b_i^\top} \Delta x + \Delta x^\top \underbrace{\check{J}_i^\top \Omega_i \check{J}_i}_{=: H_i} \Delta x \\ &= \sum_i \underbrace{c_i}_{=: c} + 2 \sum_i \underbrace{b_i^\top}_{=: b^\top} \Delta x + \Delta x^\top \sum_i \underbrace{H_i}_{=: H} \Delta x \\ &= c + 2b^\top \Delta x + \Delta x^\top H \Delta x. \end{aligned} \quad (2.56)$$

In the third step, the linearized function is differentiated with respect to the update vector Δx : $\frac{\partial F_{\text{lin}}(\Delta x)}{\partial \Delta x}$. Therefore, the derivative of the global error function is determined:

$$\frac{\partial F(\check{x} + \Delta x)}{\partial \Delta x} \stackrel{(2.56)}{\approx} \frac{\partial F_{\text{lin}}(\Delta x)}{\partial \Delta x} = 2b + (H + H^\top) \Delta x = 2b + 2H \Delta x. \quad (2.57)$$

Fourth, the root of the derivative is determined to find optimal update vector Δx^* : $\Delta x^* = \operatorname{argmin}_{\Delta x} F_{\text{lin}}(\Delta x)$. For this purpose, the minimum value of the global error function is computed:

$$\Delta x^* = \operatorname{argmin}_{\Delta x} F_{\text{lin}}(\check{x}, \Delta x). \quad (2.58)$$

The minimum is calculated by setting the previously determined derivative to zero:

$$\begin{aligned} 2b + 2H \Delta x &= \mathbf{0} \\ \Leftrightarrow H \Delta x &= -b. \end{aligned} \quad (2.59)$$

The second derivative of the linearized global error function does not depend on Δx :

$$\frac{\partial^2 F_{\text{lin}}(\check{x}, \Delta x)}{\partial (\Delta x)^2} = 2H. \quad (2.60)$$

Consequently, Δx is a minimum of the linearized global error function because H is symmetric and positive semi-definite. Therefore, by solving the linear equation 2.59 by multiplying the inverse of H , the optimal update vector is obtained. In practice, the matrix inversion is a computationally expensive operation. The solution of the linear equation can be accomplished by applying the Moore-Penrose-inverse [94], a Cholesky decomposition, QR factorization, or other methods.

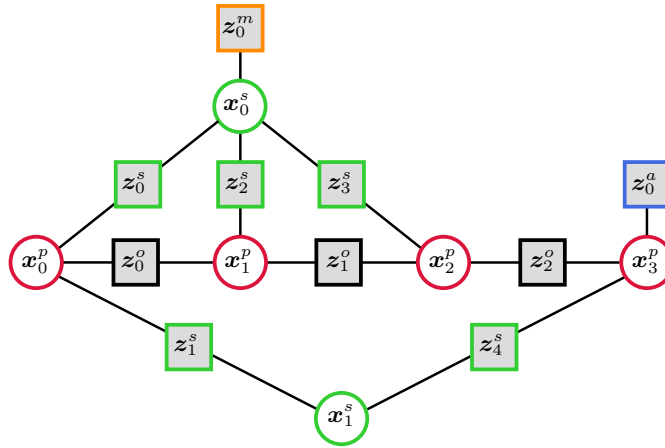


Figure 2.19: Representation of a sample factor graph with factors as square nodes and variables as circular nodes. The factor graph consists of pose variables x_i^p (red), support point variables x_i^s (green), odometry factors z_i^o (black), landmark factors z_i^s (green), map factors z_i^m (orange), and absolute pose factors z_i^a (blue). The optimization of this graph estimates the optimal states.

The last step is the update of the state estimation \check{x} as the sum of the update vector Δx and the current state estimation \check{x} :

$$\check{x}^* = \check{x} + \Delta x^*. \quad (2.61)$$

This yields the initial state for the next iteration step.

These five steps are performed until either Δx is smaller than a preset threshold or a maximum number of iterations is reached.

Factor Graph Representation: The probabilistic SLAM problem of Equation 2.43 can be represented with factor graphs [65, 160]. A factor graph is a bipartite undirected graph established by Kschischang et al. [66]. It serves as a visualization to break down probabilistic problems like SLAM, consisting of a product of functions with many variables. In these problems, the joint probability distribution is divided into multiple factors. These factors depend just on a subset of the random variables, which creates a clear decomposition of the complex probabilistic problem. The dependencies between the random variables are the measurements. The goal variables x are represented as nodes. The measurements z are represented as factors constraining the nodes.

Figure 2.19 shows a sample factor graph with the desired variables x_i , which are often illustrated as circular nodes. The variables include the robot's poses x_0^p, \dots, x_3^p and the poses of the landmarks x_0^s and x_1^s . Landmarks are distinctive elements in the environment, often called landmarks, which are used to estimate the robot's pose by detecting those characteristic objects on-board and matching them with the map landmarks. All variables together form the state vector x . As the solution approach of the SLAM problem optimizes the state vector, the poses of the landmarks are optimized as well as the robot's pose.

The measurements z_i , i.e., the factors, constrain the random variables which are connected to them by edges and square nodes in the factor graph. In Figure 2.19, multiple types of factors are shown. The absolute measurement z_0^a , which could be a Global Positioning System (GPS) pose, is only connected with one node constraining just the vehicle pose x_3^p . That means that the error between z_0^a and x_3^p should be minimal. It is therefore called unary factor. The absolute map factor z_0^m is also connected to only one node, the state vector's landmark pose x_0^s . This map factor is determined by matching

an on-board detected landmark with a map landmark. The map landmark is generated offline as a detected landmark using data capturing the environment with high precision. The corresponding pose of the map landmark forms the map factor. It is therefore only relevant for the state vector's matched, corresponding on-board landmark, here x_0^s . In this case, the difference between z_0^m and x_0^s is represented by the constraint.

The odometry measurements z_i^o and the on-board measurements of the landmarks z_i^s are relative measurements. They connect two nodes and thus are called binary factors. Odometry measurements, capturing the relative movement of the vehicle, constrain two consecutive robot poses. I. e. the error between the odometry and the difference between two consecutive robot poses should be minimized. Relative landmark measurements link robot poses with the landmark pose of the state vector. A landmark can be measured on-board from multiple robot poses. E. g. the state vector's landmark x_0^s is connected to the poses x_0^p and x_1^p . The difference between the measured landmark pose and the difference between the state vector's landmark pose and the robot's pose should be minimized.

The main problem of the Graph-based SLAM is the association of data. Here, one has to identify relations between landmarks measured at different times with respect to the uncertainties of the measurements. This is done to determine which landmarks, extracted on-board at the current time step, can be matched with landmarks, extracted on-board at former time steps. Additionally, the measured landmarks need to be associated with the map landmarks. It is decided which of the associated on-board landmarks, the output of the previously described step, correspond to which map landmark.

The computation time depends on the number of elements of the state vector while optimizing the factor graph for solving the SLAM problem. Applying the factor graph in real-time, one new pose with corresponding measurements is added every time step. Simultaneously, the number of elements of the state vector increases which increases the computational time of optimizing the graph and the complexity of the problem. Therefore, the sliding window approach is used to restrict the computational time. It limits the number of poses in the factor graph to a maximal threshold. If that threshold is reached, the oldest pose with its measurements is removed from the factor graph [160].

Part II.

Describing and Selecting Features

3. LiDAR-Based Descriptors

The review of the related work in Section 2.1 illustrates the relevance of LiDAR-based descriptors. Their definition is the main task of the *LiDAR-Feature-based Localization's* first step. Several research works already introduced methods to describe local neighborhoods within the point cloud expressively. Often, the focus is not set on real driving scenarios. Consequently, the problem of the right choice of the local neighborhood size is neglected. This chapter focuses on the research topics identified in the literature review of Subsection 2.1.4. This chapter's methodology, including the assessment criteria of descriptors and the derived research questions, is presented in Section 3.1. Based on these criteria, several existing descriptors, geometry- and intensity-based descriptors, are evaluated and extended in real driving scenarios, refer to Section 3.2 and Section 3.3. Additionally, novel descriptors overcoming the issues of the existing methods are presented in Section 3.3.

3.1. Terms and Methodology

The following section presents the methodology of the descriptors' evaluation and further developments. Therefore, the criteria for defining an expressive descriptor are established. The research questions derived from the literature review are introduced, which are answered later in this chapter. References to related publications resulted from this work are given where they are applicable.

3.1.1. Defining the Description Computation

In this subsection, the terms of the first step of the *LiDAR-Feature-based Localization* are defined. The first step is to characterize patterns within a local neighborhood around individual points in the LiDAR sensor's point cloud. A description defines these patterns.

Therefore, the local neighborhood must be defined.

Definition 3.1 (Local Neighborhood). Let $\mathcal{P} = (\mathcal{P}^{x,y,z}, \mathcal{P}^i) \subset \mathbb{R}^3 \times [0, 1]$ be a measured point cloud with 3D and intensity information and let $m : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}_{\geq 0}$ be some metric. Then, the local neighborhood $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ with radius $r \in \mathbb{R}_{>0}$ around the j -th point $\mathbf{p}_j^{x,y,z} \in \mathbb{R}^3$ is defined by

$$\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}) = \{(\mathbf{p}^{x,y,z}, \mathbf{p}^i) \in \mathcal{P} = (\mathcal{P}^{x,y,z}, \mathcal{P}^i) \mid m(\mathbf{p}^{x,y,z}, \mathbf{p}_j^{x,y,z}) \leq r\}.$$

The term $\mathcal{N}_{\mathcal{P},r}$ replaces the term $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ in cases where no risk of confusion exists. A local neighborhood is called spherical when the metric m induced by the l^2 -norm is used.

The definition of the local neighborhood enables the definition of the descriptor function.

Definition 3.2 (Descriptor Function). Let $r \in \mathbb{R}_{>0}$ be fixed. The descriptor function, or short descriptor, is a function

$$D : \{\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}) \mid \mathcal{P} \subset \mathbb{R}^3 \times [0, 1], \mathbf{p}_j^{x,y,z} \in \mathbb{R}^3\} \rightarrow \mathbb{R}^m$$

mapping a local neighborhood $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ around the j -th point $\mathbf{p}_j^{x,y,z} \in \mathbb{R}^3$ to an element in \mathbb{R}^m .

The descriptor function aims to compress the information contained in a local neighborhood without losing relevant information.

After this definition, a description can be defined.

Definition 3.3 (Description). Let $r \in \mathbb{R}_{>0}$ be fixed and let $D : \{ \mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}) \mid \mathcal{P} \subset \mathbb{R}^3 \times [0, 1], \mathbf{p}_j^{x,y,z} \in \mathbb{R}^3 \} \rightarrow \mathbb{R}^m$ be a descriptor function mapping $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ around the j -th point $\mathbf{p}_j^{x,y,z} \in \mathbb{R}^3$ to an element in \mathbb{R}^m . Then, the description $\mathbf{d} \in \mathbb{R}^m$ of point $\mathbf{p}_j^{x,y,z} \in \mathbb{R}^3$ is defined by

$$D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})) = \mathbf{d}.$$

For a local neighborhood $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ the term $D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}))$ is called the description of \mathbf{p}_j . The set of descriptions of all points in a point cloud \mathcal{P} is denoted with $\mathcal{D}_{\mathcal{P}}$.

Thus, the goal of the first step of the *LiDAR-Feature-based Localization* can be formulated as follows.

Problem Formulation:

Let $\mathcal{P} = (\mathcal{P}^{x,y,z}, \mathcal{P}^i) \subset \mathbb{R}^3 \times [0, 1]$ be a measured point cloud with 3D and intensity information. The goal is, for a given descriptor D , for each point $\mathbf{p} \in \mathcal{P}$, and for neighborhood radii $r \in \mathbb{R}_{>0}$, to determine descriptions $\mathcal{D}_{\mathcal{P}}$ such that the computed descriptions are good.

3.1.2. Procedure of Descriptor Evaluations and Extensions

The procedure of the descriptor analyzes consists of two steps: the selection and the evaluation of descriptors. It is based on the conclusions drawn from the relevant literature, see Subsection 2.1.4.

Selection of Descriptors: Several descriptors are selected from the literature. For this purpose, the division of the key descriptor types in Subsection 2.1.1 is taken into account, refer to Figure 2.1. While some categories are omitted, from each of the remaining categories, some descriptors are picked.

This thesis concentrates on handcrafted descriptors to narrow down the topics that are studied in this thesis, refer to Subsection 2.1.1. In this work, a model-based as opposed to an AI-based approach is pursued. On the one hand, model-based processes are more comprehensible than AI-based ones and their verification and validation has not yet been resolved for AI-based procedures. On the other hand, the motivation of the work is based, among other things, on the fact that localization should not only be possible in predefined scenarios, but can be generalized. This is in contradiction to available training data. This effect becomes problematic as soon as the domain of the training data is left in the practical application of neural networks. Since the geometric properties of the point clouds can be described conventionally, a model-based method is used in this work. Therefore, AI-based descriptors are not included in the analyzes of this chapter. Nevertheless, they were shortly examined during the time of this thesis. In appendix A, an outlook of these examinations is depicted. The results show that the trained networks do not achieve generalization of the training data. This is not an indication that AI-based approaches are fundamentally unsuitable for use. However, it doesn't seem easy to apply.

Also, the concept of detector-based descriptors are neglected in this thesis. Detector-based methods only describe single salient points, which are extracted beforehand based on simple characteristics. This preselection is not based upon characteristics for localization, which is advisable applying the descriptors in this use case. For example, those points of a LiDAR point cloud are often kept for the describing step which have high depth changes in their neighborhood. Nevertheless, points with small depth changes, like walls, provide additional information for localization, i.e., knowledge about the lateral position. Additionally, as described in Subsection 2.1.4, the use of single key points in localization is not suitable. A repeatable matching of single key points of sparse point clouds is hardly possible. Consequently, in this thesis, the concept of detector-based algorithms is disregarded. However, the

principle of the describing part of the detector-based MeshHOG descriptor is applied to develop a novel descriptor [167].

For those reasons, this thesis considers mostly descriptor-based methods. Descriptor-based methods are divided into two groups, unorganized structures and organized structures. Unorganized descriptors describe local neighborhoods of point clouds in a straightforward way without preprocessing steps, which would make the algorithm more robust. This approach is easy to implement but is very sensitive to noise, occlusion, and clutter. Therefore, their application to real-world data is not desirable and not considered part of this thesis. Nevertheless, some experiments on unorganized descriptors can be seen in [54], which was published during the work for this thesis.

Of the group of organized structures, all subgroups are considered. Two descriptors are selected from the geometric descriptors subgroup. The popular FPFH descriptor is chosen, an improved algorithm of the PFH [108–110], refer to Subsection 2.1.1. As a second popular geometric descriptor, the SHOT descriptor is chosen [134], refer to also Subsection 2.1.1. Further, a spatial descriptor, the RoPS by Guo et al., is considered in this thesis [43, 45]. Descriptors from both categories process the x , y , and z data of the LiDAR point cloud, and hence, their analyzes as geometry-based descriptors are summarized in Section 3.2.

Besides, intensity-based descriptors are investigated. The potential of intensity-based descriptors has not been analyzed completely yet. Two of a few approaches have been summarized in [20, 42] by Cop et al. and Guo et al., refer to Subsection 2.1.1. These descriptors, called DELIGHT and ISHOT, are examined in theory.

In total, five descriptors are chosen from the literature, the FPFH, the SHOT, the RoPS, the DELIGHT, and the ISHOT descriptor.

Evaluation of Descriptors: The selected descriptors should be evaluated for their application in the *LiDAR-Feature-based Localization*. Therefore, several criteria must be specified, defining a good descriptor applicable in the context of this thesis.

Definition 3.4 (Good Descriptor). A descriptor D is called good if there is a function $R(\mathcal{P}, \mathbf{p}) = r$ computing a radius and a descriptor-specific metric $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ such that the following conditions are fulfilled:

- (i) Let $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ be a local neighborhood, then $D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}))$ has to describe characteristic relations between sampling points enclosed by the local neighborhood in the context of localization.
- (ii) Let $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ and $\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z})$ be local neighborhoods enclosing points of a similar object and let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some metric, then $m_d(D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})), D(\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z}))) \approx 0$.
- (iii) Let $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ and $\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z})$ be local neighborhoods enclosing points of dissimilar objects and let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some metric, then $m_d(D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})), D(\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z}))) \gg 0$.
- (iv) Let $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ and $\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z})$ be local neighborhoods around the same point $\mathbf{p}_j^{x,y,z}$ enclosing points of the same object with point clouds \mathcal{P} and \mathcal{P}' of different densities and let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some metric, then $m_d(D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})), D(\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z}))) \approx 0$.
- (v) Let $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ and $\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z})$ be local neighborhoods around the same point $\mathbf{p}_j^{x,y,z}$ enclosing points of the same object with point clouds \mathcal{P} and \mathcal{P}' of different viewing angles and let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some metric, then $m_d(D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})), D(\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z}))) \approx 0$.
- (vi) Let $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ and $\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z})$ be local neighborhoods around the same point $\mathbf{p}_j^{x,y,z}$ enclosing points of the same object sampled from different distances with point clouds \mathcal{P} and \mathcal{P}' and let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some metric, then $m_d(D(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})), D(\mathcal{N}_{\mathcal{P}',r}(\mathbf{p}_j^{x,y,z}))) \approx 0$.



Figure 3.1: Sensor setup, architecture, and image of the test vehicles with five LiDAR sensors (red), four cameras (blue), a low-cost GPS (pink antenna), a localization reference system with two antennas (orange), three PCs (green), and an odometry module (gray).

This definition, including the six criteria, is explained in more detail in the following. The first Requirement (i) is strongly linked to the local neighborhood radius r . If the radius is too small, the local neighborhood does not contain information about the environment's 3D structure. If the radius is too large, the local neighborhood encloses multiple objects. Thus, descriptions often capture an ambiguous superposition of these characteristics. In both cases, a descriptor function may not describe expressive characteristic relations suitable for the localization process. Hence, the local neighborhood choice, i.e., the radius r , is an essential task. Moreover, descriptions of points on similar physical objects should be approximately the same, refer to criterion (ii), and the descriptor function should be insensitive to moderate changes of the point density of the point cloud, referring to criterion (iv). This means that a point's description must be similar despite whether sparse or dense point clouds are being used. This is, for example, important because a descriptor function should be able to be applied independently of the sensor type. However, the descriptions of points on dissimilar objects should be very dissimilar, refer to Requirement (iii). The last two requirements (v) and (vi) state that a point's description should be both insensitive to moderate changes of the sensor's viewing angle and of the distance between the sensor and the point. For example, the description of a point on a wall should be approximately the same regardless of the vehicle's distance to the wall and of the vehicle's orientation.

These criteria are used in this second step to evaluate the selected descriptors. For each of the criteria, theoretical or practical experiments are designed and performed in the following. The experiments' results are used to evaluate the selected descriptors. Based on these results, some descriptors are extended and novel algorithms are developed to overcome the issues of state-of-the-art descriptors, which are detected in the evaluation.

Data Sets: For the evaluation, several data set types of LiDAR point clouds are used as input data for the description calculation: Synthetic data modeled after existing LiDAR sensors and real data, recorded with close-to-production-type LiDAR sensors with the test vehicle shown in Figure 3.1 and sampled and accumulated in postprocessing with a mobile data acquisition system, are considered.

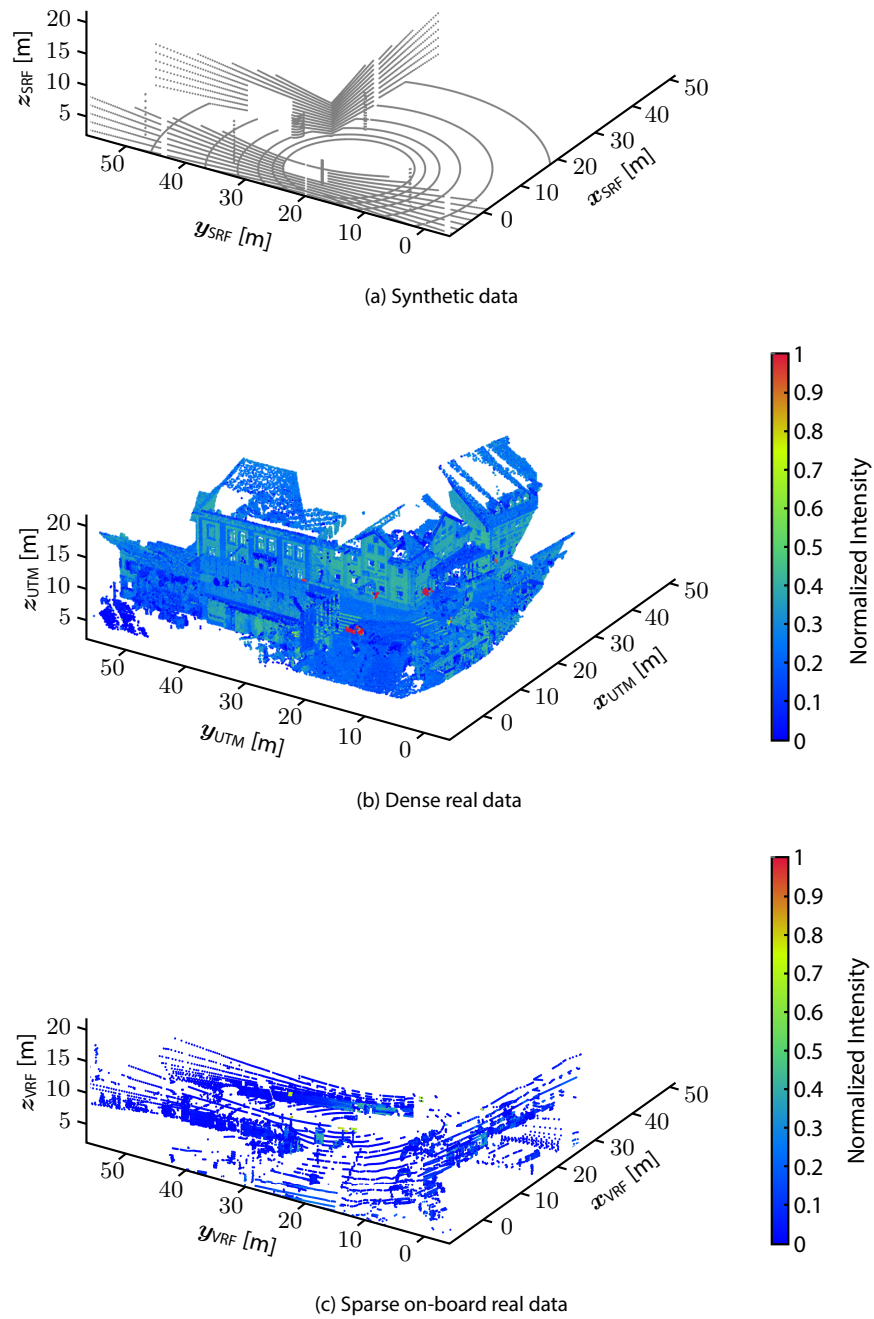


Figure 3.2: Sample LiDAR point clouds of each data set type used in this thesis in their corresponding coordinate system. The dense and sparse real data capture the same section of the environment.

Sample LiDAR point clouds of all of the three data sets are depicted in Figure 3.2. The synthetic data can be understood as ideal data, while the real data contains measurement inaccuracies and noise. Intuitively, the descriptors should yield more accurate results on synthetic data than on real data. Though, concerning the application of the descriptors in a vehicle, the experiments with real data are more important.

For the generation of realistic synthetic data, geometric objects of different size, which often occur in the environment, like planes, edges, and cylinders, are constructed. They are combined into a complete scene with walls, building edges, pillars, and advertising pillars. In this thesis, a virtual scanner is implemented to model the beams of LiDAR sensors and determines the intersections between the beams and the objects. The virtual scanner is instantiated to the specifications of the Velodyne VLP 16 [151] and the Velodyne VLP 32-C [150] LiDAR sensor generating the synthetic point cloud.

For the generation of real world point clouds, on the one hand, the test vehicles' LiDAR sensors on the front left side, i.e., the Velodyne VLP 16 [151] and the Velodyne VLP 32-C [150], are used to create the on-board real-world point clouds, see 3.1. As the names of the sensors state, the first sensor has 16 diodes and the second sensor has 32 diodes. The measurements in one horizontal direction of one diode form a layer, the measurements in one vertical direction of all diodes lie in one channel. The Velodyne VLP 16 samples the environment with 2° in vertical and 0.2° in horizontal resolution. It has a vertical field of view of -15° to 15° , which makes in total an opening angle of 30° . The Velodyne VLP 32-C samples the environment with approximately 9° to 0.333° in vertical and 0.2° in horizontal resolution, i.e., with a non-uniform vertical resolution, see Figure 3.2. It has a vertical field of view of -15° to 25° , which makes in total an opening angle of 40° . The sensors provide data with 10 Hertz. The on-board point clouds represent one LiDAR scan, i.e., rotation of the sensor, either in the local SRF or transformed into the local VRF with high-precision calibration data, whenever needed.

On the other hand, the dense point clouds are given in the global coordinate system UTM. In the figures of this thesis, the UTM coordinates are shifted to an origin lying in the point cloud's range. These point clouds are collected by the mobile data acquisition system Trimble MX8 [105, 142], accumulating, postprocessing, and globally referencing the point clouds. It yields highly accurate measurements. The resulting sampling resolution of the dense data is approximately $34 \frac{\text{points}}{100 \text{ cm}^2}$ and its point cloud is assumed to be equidistant. These accurate point clouds are later used as a basis to create the map for localization as they represent the real world nearly ideally.

The same excerpt of a real-world scenery is shown in figures 3.2b and 3.2c. It can be seen that both point clouds look very different even though they depict the same section. Hence, it is difficult to similarly interpret both data set types with a descriptor, thus yielding similar descriptions, to fulfill Requirement (iv). However, this is important to handle so that the on-board data, sparse point clouds, can be matched with the map data, dense point clouds, for localization.

However, the test vehicles are equipped with additional sensors, whose data is used in this thesis. The vehicle's odometry is measured with a software module called EgoMaster [3]. The odometry is calculated in the VRF, refer to Subsection 2.2.1. It is a software module, computing the vehicle's odometry based on data from standard and close-to-production type sensors integrated into the Electronic Stability Program (ESP) and Anti-Lock Braking System (ABS). The sensors are angular rate sensors, steering angle sensors, wheel speed sensors, chassis lift sensors, and IMUs. The EgoMaster fuses the data received from the sensors applying an EKF to compute the odometry. The odometry data is provided with 100 Hertz. GNSS data is received with a standard production-type GPS sensors. In this thesis, the NovAtel FlexPak-G2 OEMStar [91] is mounted on the test vehicle including an antenna receiving GPS and GLOBal Navigation Satellite System (GLONASS) signals. In the test drives of this thesis, the data rate is set to 5 Hertz. Position reference data is gained with the high-precision Applanix POS LV 520 system [1] with a data rate of 200 Hertz. For a more accurate estimation of the heading, two antennas are mounted on different lateral positions of the vehicle receiving RTK-GPS data. The data of the inte-

grated IMU is applied in postprocessing estimating a highly-precise vehicle trajectory (0.02 meters in x , and y direction, 0.015° for the heading).

All of the sensor data is logged and passed on on three PCs in the test vehicles.

3.1.3. Research Questions of the Application of Descriptors

Below, the research questions addressed in this chapter are listed and explained in more detail. An overview of the papers, which were published during the course of this thesis and which investigate this research questions, is given thereafter.

Are geometry-based descriptors well-suited for a real-world localization?

- As explained in Subsection 2.1.4, the performance of descriptors largely depends on the data they are applied to. The main difficulties in this thesis when dealing with real data lie in the sparsity and the noisiness of on-board LiDAR point clouds.
- Selected algorithms are applied to all data sets, either modeling or capturing a typical environment, to evaluate geometry-based descriptors' suitability. The assessment of descriptors is based on several evaluation parameters, refer to Definition 3.4, like the distinction between objects, the dependence on the range, and the dependence on the local neighborhood size. Real data of on-board sensors are used to compute odometry and global localization accuracies.

Are intensity-based descriptor algorithms as useful for localization as geometry-based ones?

- Intensity information of LiDAR point clouds is just one-dimensional compared to the correlating three-dimensional geometric data. Thus, it does not provide as much information for a descriptor function.
- State-of-the-art descriptors are evaluated in theory and a novel descriptive descriptor is developed to estimate the suitability of intensity-based descriptors in Section 3.3. The development's focus lies on the challenging problem of handling real intensity data. Then, the concept of intensity-based descriptors is evaluated by comparing it to geometry-based descriptors and applying both descriptor types to a simple odometry and localization algorithm to real-world data collected with LiDAR sensors on-board test vehicles.

During the course of this work, two publications examining the preceding research questions were made.

In [54], the findings for several evaluation parameters of the first research question are presented. The accuracy with which road objects can be recognized by their descriptions for localization tasks depending on the distance and direction of view of the sensor is shown using synthetic and real data. Here, the FPFH descriptor and unorganized structured descriptors, see Equation 2.1, are applied. Furthermore, the accuracy allocating an object only based on its descriptions is determined on the example of a building corner. As a result of these investigations into the description calculation, conclusions about the localization accuracy that can be achieved are drawn.

The developed intensity-based descriptor is introduced and compared to geometry-based descriptors in [52], answering the second research question. The intensity-based descriptor is called GRAdients of Intensities as a Local descriptor (GRAIL). The position accuracies of the GRAIL applying a global localization with respect to a map are compared to the accuracies achieved with the FPFH, RoPS, and SHOT descriptors. This has been thoroughly examined on a real-world data set.

3.2. Geometry-Based Descriptors

The algorithms selected in Subsection 3.1.2, which process only the geometry information of the LiDAR point cloud, are evaluated in this section. Following the findings of these evaluations, further algorithms are developed.

3.2.1. Analyzes of Geometry-Based Descriptors in Real-World Environment

The FPFH, the SHOT, and the RoPS descriptor is evaluated based on Definition 3.4. Each descriptor is assessed with experiments addressing every criterion of this definition. In the following, the experiments and their results are depicted, referring to the tested criterion.

1. Influences of the Local Neighborhood Size:

In the following experiments, the impact of the local neighborhood size on the descriptions is evaluated. These experiments inspect the Requirement (i) of the definition of a good descriptor 3.4. Therefore, two experiments are presented. The first experiment addresses the effect of non-uniform sampling on the choice of the local neighborhood's radius. The second experiment deals with the influence of the local neighborhood size on the description calculation outcome. In both experiments, only the metric m of the local neighborhood Definition 3.1 induced by the maximum-norm is used.

Requirement (i) of Definition 3.4 states that the descriptor should enable a representation of the three-dimensional characteristic of an object within a local neighborhood, i.e., the right choice of the local neighborhood radius. Therefore, the following definition is introduced:

Definition 3.5 (Large Enough Radius of a Local Neighborhood). Let $\mathbf{p}_{m_1}, \dots, \mathbf{p}_{m_n} \in \mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ be points of the local neighborhood in layers with numbers $l_{m_1}, \dots, l_{m_n} \in \mathbb{N}$, respectively. A radius r of a local neighborhood $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ around point \mathbf{p}_j in layer with number $l_j \in \mathbb{N}$ of an ordered point cloud is called large enough if at least three layer numbers of $\{l_j, l_{m_1}, \dots, l_{m_n}\}$ are different. If exactly three layer numbers are different the radius is called minimal.

This definition of the large enough neighborhood radius r ensures that the point environment's three-dimensional structure is characterized within a large enough local neighborhood. If the local neighborhood encloses less than three layers, the local neighborhood's line characteristic dominates the description values. If the local neighborhood encloses at least three layers, the three-dimensionality, x , y , and z , can be described in the local neighborhood. Therefore, the radius of local neighborhood is large enough and its neighborhood must enclose at least three layers to enable the description to capture the three-dimensional characteristic.

First experiment: In the first experiment, typical sampling characteristics of close-to-production LiDAR sensors are considered to examine the influence of the local neighborhood size. Therefore, two sensors are examined, the Velodyne VLP 16 and the Velodyne 32-C, which are used throughout this thesis, but reflect typical problems with on-board LiDAR sensors. The main problem of real on-board sensors concerning the local neighborhood size is caused due to the different horizontal resolution of these sensors compared to the vertical resolution, refer to Subsection 3.1.2. This affects the eigenvalue decomposition many descriptors depend upon, refer to Subsection 2.1.1.

Figure 3.3 shows the eigenvalues computed with a Principal Component Analysis (PCA) with respect to the local neighborhood size around a point on the middle of a synthetic vertical plane modeled for both sensor types. See Subsection 2.2.2 for an explanation of the PCA. As the plane is two-dimensional, only two eigenvalues are depicted, with $\lambda_1 > \lambda_2$. The blue eigenvalues represent the eigenvalues of the vertical plane sampled with the VLP 32-C, and the orange eigenvalues correspond to the VLP 16. The red line visualizes the reference eigenvalues of a uniformly sampled plane with a high resolution.

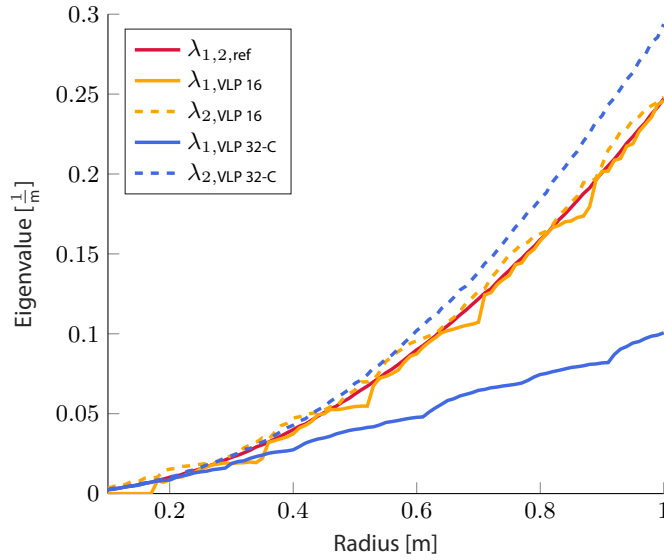


Figure 3.3: Eigenvalues of a synthetic vertical plane sampled with the Velodyne VLP 16 and the VLP 32-C as a function of the radial local neighborhood radius around a point in the middle of the plane

The eigenvalue λ_2 is represented by a dashed line, the smaller with a solid line. For a uniformly sampled plane, both eigenvalues are the same, visible with the red line. The eigenvalues of the VLP 16 show that the point cloud's sparsity leads to discretization effects even if the horizontal layers are equally distanced. Each time that the local neighborhood encloses one additional layer, a significant change of the two eigenvalues can be seen. For differently distanced layers, the eigenvalues drift apart with larger local neighborhood radii. This experiment shows that the local neighborhood size distorts the eigenvalues in case of on-board, close-to-production-type LiDAR sensors, which are the basis for many descriptors. This may lead to description inaccuracies, as often the LRF of a description is calculated using eigenvalues. The effect becomes especially obvious, looking at unstructured geometry-based descriptors, refer to Subsection 2.1.1 or see Equation 2.1. As the eigenvalues of a planar wall are no longer the same in case of non-uniform sampling, the linearity and the planarity are not zero and one even for synthetically, noise-free data.

Second experiment: In the second experiment, the description differences, depending on the local neighborhood radius, are examined. The experiment is divided into two parts, a qualitative and quantitative evaluation. The first part examines the absolute descriptions for each local neighborhood radius, while the second part considers the relative changes of the descriptions.

For the first part, real data from measuring a pillar is the input for the description calculations. The same pillar is sampled with the VLP 16 sensor and also extracted from the dense point cloud data. Hence, this experiment also explores the difference between differently sampled data, refer to Requirement 3.4 (iv). The descriptions are computed with a local neighborhood radius from one centimeter, in case of the dense point cloud, and from 10 centimeters, in case of the sparse point cloud, to one meter for every centimeter.

Figure 3.4 illustrates the histograms of the FPFH descriptor of these calculations. The 11 bins of the three angles θ , α , and φ are presented independently. Each bin of the 11 bins is marked in the same color, while the 11 bins are stacked for every description of each local neighborhood. It can be seen that the description changes a lot with increasing local neighborhood radii. With a small local

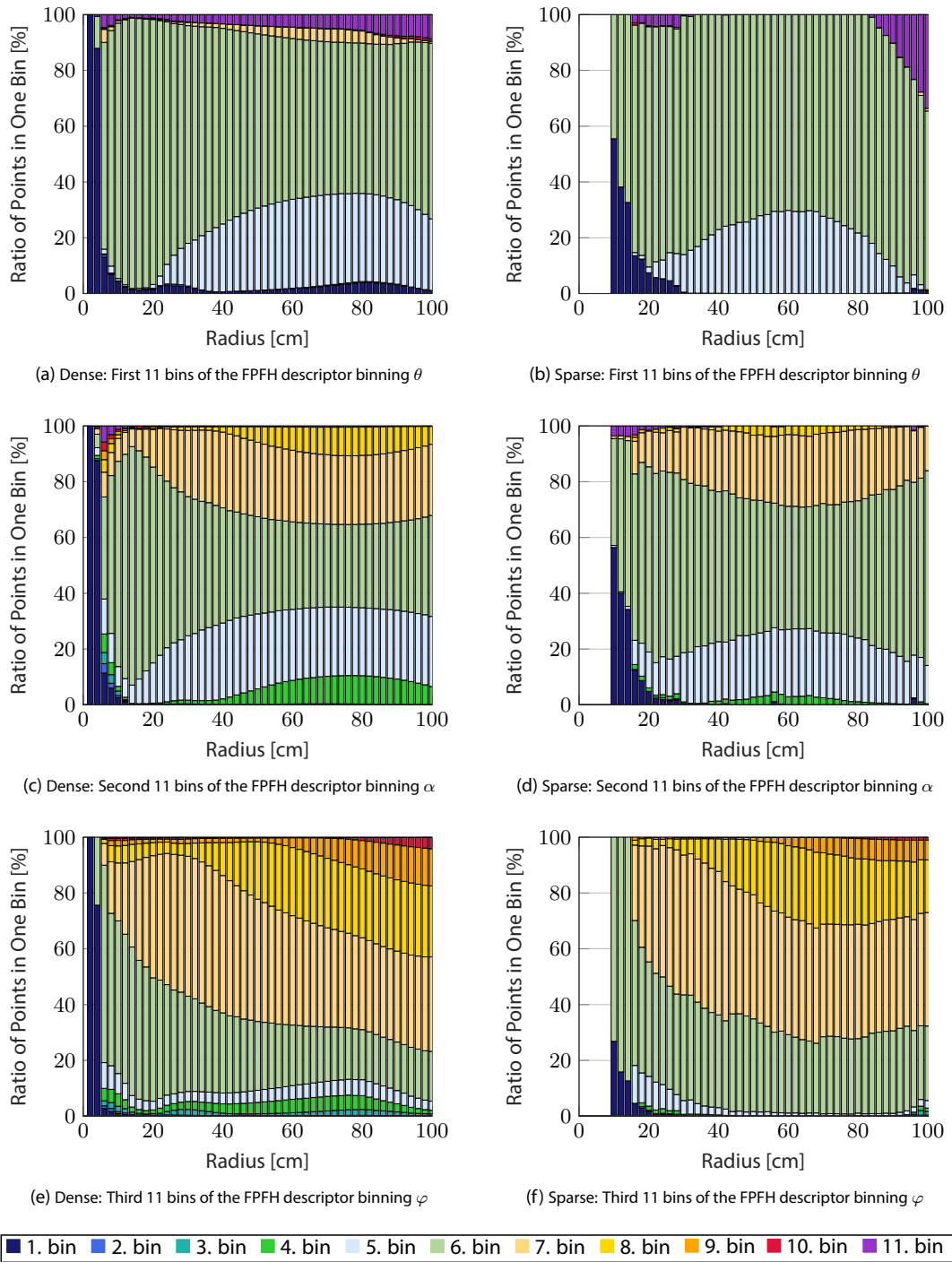


Figure 3.4: The ratio of points in a histogram bin as a function of the local neighborhood radius for every angle of the FPFH description in a stacked bar using the example of a real data pillar of radius 20 cm: dense (left) and sparse (right)

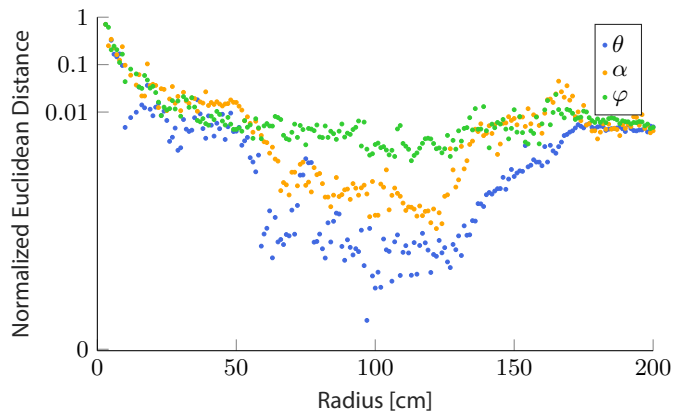


Figure 3.5: Normalized Euclidean distance as a function of the local neighborhood radius in a logarithmic scale for every angle of the FPFH description using the example of a dense real-data wall

neighborhood radius up to 20 centimeters, the description does not capture the pillar’s curvature, neglecting the effects where the local neighborhood encloses only a few points. As the neighborhood is so small, these points in it lie almost on a plane. Thus, the description looks like that of a plane object with no curvature. This results in a FPFH description with a high percentage of points for θ in the first bin and the other angles in the middle bin. With a larger local neighborhood, the points in the local neighborhood form the curved pillar. Thus, the description depicts a curved geometry with more points with different angles falling into different bins, i.e., essentially in the fifth, sixth, and seventh bin. This is true for both the dense and the sparse point clouds, while the local neighborhood radius needs to be larger for sparse point clouds to enclose and capture the pillar’s curvature. This is the case since there are less points in local neighborhoods of the sparse point cloud. The description graphs look very much alike except that the graph generated with the dense point cloud looks smoother, which could be expected.

The same holds for the other selected descriptors, except that the description representing the pillar starts with a much larger local neighborhood as they divide the neighborhood into multiple parts for all three dimensions, refer to Subsection 2.1.1. According to Definition 3.5, every part of the local neighborhood must enclose enough points to be minimal. Therefore, to capture the 3D information with the description, the local neighborhood is larger. Hence, the total local neighborhood size must be larger in partitioned local neighborhoods than non-partitioned ones. A resulting disadvantage is that this larger local neighborhood enclosing more points increases the computational effort. Besides, LiDAR sensors sample the surface of the objects in the environment. Thus, a partition of the local neighborhood in the direction of the depth is not beneficial, in general.

This part of the second experiment indicates with qualitative analyzes that the right choice of the local neighborhood is essential to get a characteristic description of an object. This statement is true for every type of descriptor, while the local neighborhood which subdivide the local neighborhood into partitions needs to be larger to capture the characteristics. It also suggests that the descriptions of sparse and dense point clouds are nearly identical, even in their dependence on the local neighborhood size.

The second part of this experiment underlines the effect that a minimum local neighborhood must be reached to measure an object’s characteristic description by analyzing the relative change of the descriptions. In this example, a dense real data wall is examined again on the example of the FPFH

descriptor. The FPFH description is calculated for every local neighborhood radius starting from one centimeter to two meters. Here, the normalized Euclidean distance $m_{n \text{ eucl}}^{\text{FPFH}}$ is used to measure the change of the 33-dimensional descriptions d^{FPFH} between two local neighborhood radii:

$$m_{n \text{ eucl}}^{\text{FPFH}} = \frac{1}{\sqrt{2} \cdot 3} \cdot \sqrt{\sum_{n=1}^{N=33} (d_n^{\text{FPFH},1} - d_n^{\text{FPFH},2})^2}, \quad (3.1)$$

where $d_n^{\text{FPFH},i}$ denotes the n -th entry of the i -th description. The FPFH description consists of three to $[0, 1]$ normalized histograms for each angle θ, α, φ . For each histogram, the maximal difference between two different descriptions is 2. Thus, for the three concatenated histograms, i.e., the FPFH description, the normalization factor is $\frac{1}{\sqrt{2} \cdot 3}$.

Figure 3.5 shows the normalized Euclidean distances between two FPFH descriptions calculated with increasing neighborhood radius for each of the three angles of the description independently. It can be seen that the relative changes for every angle of the FPFH description are large when the local neighborhood is small. At about 25 centimeters, the normalized Euclidean distance is relatively small and does not change until it increases at approximately 160 centimeters. The reason for this is that the characteristic description of the wall is reached with approximately 25 centimeters, as there are enough points in the local neighborhood forming a wall. Additionally, the point cloud's noisiness decreases in the description of the FPFH due to the averaging the more points are enclosed in the local neighborhood. This effect influences the size of the local neighborhood to reach the characteristic description of an object. The descriptions stay the same until the local neighborhood encloses points of other objects at 160 centimeters. At this radius, the normalized Euclidean distance is smaller than the Euclidean distances up to 25 centimeters, because more points are located on the wall than on the newly enclosed object. I. e. the ratio of the number of points on the wall to the number of points on the other object is large. Thus, the change of the FPFH histogram is relatively small.

Repeating this experiment with other descriptors yields similar graphs, but the description changes are smaller when the local neighborhood encloses other objects' points. The same reason as above applies here as well. The division of the local neighborhood into multiple parts leads to smaller segments of the local neighborhood, which still must enclose enough points to capture the three-dimensional characteristic of the newly enclosed objects.

In this second part of the experiment, the quantitative analyzes show the relative changes of the descriptions depending on the local neighborhood radius. It can be seen that a minimum local neighborhood radius must be chosen to compute the characteristic description of an object sampled by on-board LiDAR sensors. This part also demonstrates that the description changes again when points of other geometric objects are enclosed by the local neighborhood, indicating a maximum local neighborhood for an object in the environment.

Summarizing, these experiments show that the size of the local neighborhood affects the descriptions a lot. With the discrete sampling of the environment by LiDAR sensors, the eigenvalue computation, which is the basis for many LRFs, is affected depending on the local neighborhood radius. This effect is more visible in case of non-uniformly distributed layers. The second experiment shows that a specific minimum local neighborhood size must be chosen to characterize an object distinctively, while a too-large local neighborhood might enclose points belonging to other objects, which also causes changes in the description.

2. Distinguishing Between Objects:

Requirements (ii) and (iii) of Definition 3.4 are examined for the FPFH, SHOT, and RoPS descriptor in this paragraph. They state that a description of unlike geometric objects should be dissimilar and

similar for alike geometric objects. Therefore, three different types of objects are picked, often used as landmarks for a semantic localization: a wall, a pillar, and a vertical edge [14, 119, 123, 160]. The descriptions of these objects are compared to each other in a distance matrix.

A description of an object is defined as medoid of all descriptions of the points of the object. Therefore, the medoid is defined.

Definition 3.6 (Medoid). Let $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ be a set containing n elements in a space and let m be a distance function. The medoid \mathbf{a}_{medoid} is defined as

$$\mathbf{a}_{medoid} = \underset{\mathbf{a}' \in \mathcal{A}}{\operatorname{argmin}} \sum_{i=1}^n m(\mathbf{a}', \mathbf{a}_i).$$

In this experiment, the three objects are sampled synthetically using the Velodyne VLP 16 model, and they are recorded on-board with the actual LiDAR sensor. When generating the synthetic point cloud, the wall is modeled as a planar, vertical surface, the pillar is modeled as a cylinder, and the vertical edge is modeled by two planes perpendicular to each other. The descriptions are calculated for every point of the object and their computed medoid. The medoid represents the whole object the best. The description distances are calculated with normalized Euclidean distances. As the normalization is different for every descriptor type, the following formulas for the (normalized) Euclidean distances $m_{(\text{norm})\text{eucl}}$ are introduced, where d_n^i represents the n -th entry of the i -th description:

$$\text{FPFH: } m_{n \text{ eucl}}^{\text{FPFH}} = \frac{1}{\sqrt{2} \cdot 3} \cdot \sqrt{\sum_{n=1}^{N=33} (d_n^{\text{FPFH},1} - d_n^{\text{FPFH},2})^2}, \quad (3.2)$$

$$\text{SHOT: } m_{n \text{ eucl}}^{\text{SHOT}} = \frac{1}{\sqrt{2} \cdot 32} \cdot \sqrt{\sum_{n=1}^{N=352} (d_n^{\text{SHOT},1} - d_n^{\text{SHOT},2})^2}, \quad (3.3)$$

$$\text{RoPS: } m_{n \text{ eucl}}^{\text{RoPS},i} = \sqrt{\sum_{n=1}^{N=27} (d_n^{\text{RoPS},1} - d_n^{\text{RoPS},2})^2}, \quad \forall i = 1, \dots, 5. \quad (3.4)$$

The FPFH description vector is 33-dimensional and consists of three components, the angles θ , α , and φ . This leads to $N = 33$. As described in the previous experiment this yields the normalization factor of $\frac{1}{\sqrt{2} \cdot 3}$. The SHOT description vector is 352-dimensional and consists of 32 to $[0, 1]$ normalized histograms, the 32 parts of the local neighborhood. This leads to $N = 352$. Like for the FPFH descriptor, the maximal difference between two different descriptions is 2 for each histogram. Thus, for the 32 histograms, the normalization factor is $\frac{1}{\sqrt{2} \cdot 32}$. In the case of the RoPS descriptor, the individual moments and the entropy are examined separately. The RoPS has in total 135 description entries. These are spread over 27 description entries, each for the four moments and the entropy. For each of the five description components (four moments and the entropy), the Euclidean distance is calculated from the 27 descriptions, which leads to $N = 27$ for every description component. Here, normalization is not performed, as the RoPS describes absolute moments and the entropy, which are not relative as for the FPFH and the SHOT histogram-based descriptors. Normalization with the maximum descriptions is also not carried out since the maximum descriptions differ a lot depending on the object. The best way to compare the description of the RoPS is by looking at the absolute values.

In the following, we discuss the differences between the three object classes based on the reference medoid distances of Table 3.1. For all descriptors, it can be seen that the descriptors can distinguish between the three object classes.

Table 3.1: Distance matrix of FPFH, SHOT, and RoPS medoid descriptions for a real and synthetic wall, pillar, and vertical edge specified with (normalized) Euclidean distance of Equation 3.1 sampled with Velodyne VLP 16 (model) with the same local neighborhood size of 70 centimeters

		wall	pillar	vertical edge
FPFH	wall^a	-	0.214	0.432
	pillar^a	0.214	-	0.153
	vertical edge^a	0.432	0.153	-
FPFH	wall^b	-	0.323	0.466
	pillar^b	0.323	-	0.054
	vertical edge^b	0.466	0.054	-
SHOT	wall^a	-	0.384	0.391
	pillar^a	0.384	-	0.041
	vertical edge^a	0.391	0.041	-
SHOT	wall^b	-	0.423	0.498
	pillar^b	0.423	-	0.038
	vertical edge^b	0.498	0.038	-
RoPS	wall^a	-	0.06 0.04 0.05 0.06 0.06	0.1 0.09 0.14 0.04 0.06
	pillar^a	0.06 0.04 0.05 0.06 0.06	-	0.06 0.1 0.07 0.05 0.04
	vertical edge^a	0.1 0.09 0.14 0.04 0.06	0.06 0.1 0.07 0.05 0.04	-
RoPS	wall^b	-	0.15 0.14 0.08 0.1 0.09	0.17 0.15 0.15 0.08 0.04
	pillar^b	0.15 0.14 0.08 0.1 0.09	-	0.08 0.12 0.08 0.07 0.05
	vertical edge^b	0.17 0.15 0.15 0.08 0.04	0.08 0.12 0.08 0.07 0.05	-

^a Real data.

^b Synthetic data.

In detail, Table 3.1 shows that the description of a wall differs greatly from that of a pillar and a vertical edge, cf. Requirement (iii). What is also apparent is that the distances of the descriptions of an edge and a pillar are relatively small, cf. Requirement (ii). Looking at the geometrical structure of the three objects, this becomes clear. A wall is a planar object, where all normals point in the same direction, while the edge and the pillar are more curved with larger surface normal differences. The same effect holds for the spatial point distribution, in case of the RoPS descriptor. The local neighborhood of the RoPS is divided into three 2D grids. In the 2D grid, the points of a wall fall into other grid cells. For example, in synthetic data, the differences of the SHOT description of a wall and a pillar amount to 0.423 compared to the description differences of a pillar and an edge amount to 0.038.

Additionally, the real data objects' differences are almost exclusively smaller compared to those of the synthetic data. The cause for the smaller differences is the noisiness of the real data. The description of a noisy wall results in noisy normals and more spread points. Thus, the noisy wall description resembles the pillar and the edge description. This results in a greater distinction between objects in the environment in real data.

In summary, this experiment shows on the example of three object types that the geometry-based descriptors can distinguish between the object types and thus fulfill the requirement (iii) of definition 3.4. However, due to the noisiness in real data, the distinguishability is smaller compared to synthetic data. Hence, the criterion (ii) of definition 3.4 is met for these descriptors.

3. Difference Between Sparse and Dense Data:

In this experiment, the differences of descriptions between differently sampled real data point clouds are examined. This experiment evaluates the descriptors corresponding to criterion (iv) of Definition 3.4. For one thing, this is an important requirement, as the descriptors should be accurate with any sampling resolution independent of the sensor type. Additionally, this is crucial for matching on-board and map data, a part of the third step of the *LiDAR-Feature-based Localization*. Thus, three different types of objects are chosen, which are frequently employed as semantic landmarks: walls, pillar, and vertical edges [14, 119, 123, 160].

These objects are sampled in real-world with the Velodyne VLP 16, and their corresponding point clouds are extracted from the globally referenced dense point cloud. These point clouds are the input for the description calculation applied for the FPFH, SHOT, and RoPS algorithms. Then, the medoids of these descriptions are built to get the representations of the whole objects instead of the segments of the local neighborhoods. Every medoid of the sparse point cloud is compared to the medoid of the dense point cloud using the (normalized) Euclidean distances of Equation 3.2, 3.3, and 3.4.

Table 3.2 states the differences between sparse and dense real data of a wall, a pillar, and a vertical edge. It can be seen that the differences between the sparse and dense wall and edge are small for every descriptor. This is the case since the description of a noisy wall looks similar to the description of an edge. That means that the different sampling resolutions of the real data and their different noisiness affect the description calculation for these objects. Larger differences compared to the other objects can be seen considering the pillar. Here, the different densities of the point cloud induce a difference in the descriptions. This is due to the fact that a wall does not need to be sampled with a

Table 3.2: (Normalized) Euclidean distances between sparse and dense real data for the descriptors FPFH, SHOT, and RoPS

descriptor	wall	pillar	vertical edge
FPFH	0.003	0.15	0.06
SHOT	0.02	0.20	0.02
RoPS	0.05 0.16 0.05 0.06 0.05	0.02 0.04 0.14 0.11 0.06	0.01 0.02 0.09 0.11 0.02

high resolution to build the characteristic description as it has no curvature. It is still distinctive with few points on the wall. Therefore, the planar structure can also be captured with the data from the Velodyne VLP 16 sensor. The same applies to the edge. As an edge consists of two planar walls, a few points on that edge or the two walls suffice for a characteristic description of that object. This is different in the case of the pillar. The pillar is a homogeneous curved object, where the captured description depends on the sampling resolution, which is shown in the results of Table 3.2. This effect is hard to compensate with descriptors, as there is no more information which can be processed within the algorithm. Notably, the need for compensation contradicts the claim that the algorithms should be very distinctive. I. e., if a distinctive description should capture the point cloud's characteristics in great detail, a compensation might predominate the fine structures within the point cloud. Moreover, a compensation might blur the point cloud's geometric information. In addition to the noise present in the point cloud, this compensation would enhance this effect.

Concluding, the examined descriptors are considerably independent of the sampling resolution and thus match criterion (iv) of Definition 3.4. However, they are not able to compensate for the missing information in the case of sparse point clouds entirely.

4. Dependence on Viewing Angle:

This experiment analyzes the dependence on the viewing angle of the descriptors FPFH, SHOT, and RoPS. This requirement is defined by criterion (v) of Definition 3.4. The independence on the viewing angle is crucial in the case of automated driving, as objects should be detectable from different perspectives. As a vehicle moves on mostly planar ground, this experiment only analyzes the dependence on the viewing angle in 2D. Here, three objects are chosen, which are often implemented as semantic landmarks: walls, pillar, and vertical edges [14, 119, 123, 160]. Therefore, these objects are the input for the description computation. Because this rotation is hard to record on-board a test vehicle in real data, the objects are modeled synthetically for the Velodyne VLP 16 and rotated around the height axis from 0° to 45° . The wall is modeled as a planar, vertical surface, the pillar is modeled as a cylinder, and the vertical edge is modeled by two planes perpendicular to each other. In this experiment, the Euclidean differences of the description values are determined according to Equation 3.2, 3.3, and 3.4 on page 65. Each description of a rotated object is compared with the not rotated object, i.e., 0° rotation.

This analysis shows that the descriptions do not change when rotating a wall or a pillar. The normalized Euclidean distances are always zero. As most descriptors rely on the eigenvalue computation, which is independent of the viewing angle, the algorithm itself is independent of the viewing angle. The descriptions are just dependent on the sampling of the objects. This becomes obvious, looking at the results when rotating an edge. In Figure 3.6, the description differences of the rotated edge compared to the not rotated edge are illustrated. This figure shows that the differences of the SHOT description do not change a lot, when the viewing angle is changed. The differences of the FPFH and the RoPS description vary greatly the further the edge is rotated. This results from the fact that the description of an edge rotated by 45° to the sensor represents a plane and not an edge because it can no longer scan one of the planes forming the edge. Thus, the description of an edge point represents a wall, not an edge due to the sampling. The difference of the 45° rotated edge is similar to the difference of a description of a wall and an edge, refer to Table 3.1.

In summary, it can be said that if the objects are sampled sufficiently, the eigenvalues, the basis for many descriptors can be reliably, independent of the viewing angle, determined. The statistical distribution of the points, captured by the RoPS description, is also independent of the viewing angle when the examined object is sufficiently scanned.

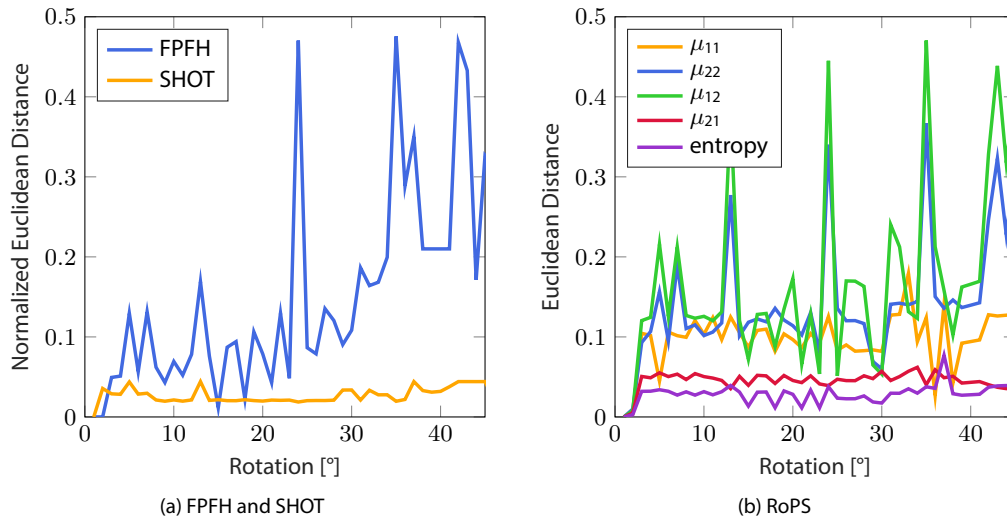


Figure 3.6: Description deviations of a rotated sparse edge point compared to the edge, pointing at the sensor, as a function of the viewing angle of the sensor on the edge point

5. Dependence on Distance:

This experiment investigates the dependence on the distance of the chosen descriptors FPFH, SHOT, and RoPS. It deals with criterion (vi) of Definition 3.4. The independence on the distance is crucial for the descriptions application for automated vehicles, as the descriptions should remain the same when approaching or moving away from an object. In particular, for localization, it is crucial to detect objects which are far away in order to estimate an accurate heading of the vehicle. Here, a pillar with a radius of 20 centimeters is taken as a sample object to examine the distance behavior, which is often used as a semantic landmark and can be detected from a large distance with this semantic prior knowledge [14, 119, 123, 160]. The recording with a LiDAR sensor of a pillar depending only on the distance is hard to realize in a test vehicle. Often, these geometric objects are not located separated from other objects or at the end of large straight street. Therefore, the pillar, modeled as a cylinder, is sampled synthetically with the Velodyne VLP 32-C model from different distances. The descriptions of the point in the middle of the sparse pillar are compared to the middle point's description of a dense pillar, which was modeled based on the dense real data, using Equation 3.2, 3.3, and 3.4 on page 65. This simulates the matching of dense and sparse data, which is important for the map matching part of localization, belonging to the third step of the *LiDAR-Feature-based Localization*.

Figure 3.7 visualizes the (normalized) Euclidean distances over the distance of five to 100 meters with a resolution of one meter for each descriptor. With increasing distance, the (normalized) Euclidean distance between the description of the sparse pillar and the reference pillar increases. This is due to the fact that the pillar is sampled more sparsely, the larger the distance to it is. The figure also shows that the deviations of all descriptions vary greatly from a distance of approximately 57.5 meters. At 57.5 meters, the pillar's curvature is no longer sampled by the LiDAR sensor, but the points are falling onto the pillar are points on top of each other, forming a straight line, not a curved cylinder. The straight line represents no longer a curved object, which results in large distances compared to a curved pillar.

Additionally, in Figure 3.7, steep increases of the Euclidean distances can be seen, before the 57.5 meters mark is reached, e.g., at approximately 29 meters. These increases occur, where the distance of the sensor to the pillar has increased that much so that no point of one certain channel is located

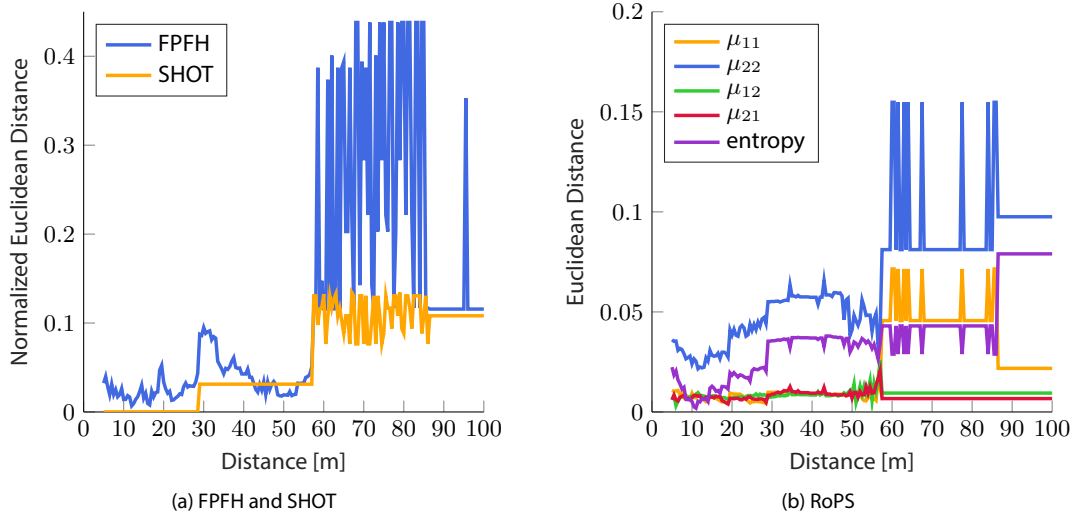


Figure 3.7: Description deviations of a sparse, synthetic pillar compared to a dense, synthetic pillar as a function of the distance to the pillar

on the pillar, e.g., instead of points of five channels, points of four channels are on the pillar. From a certain distance, the point distances in the vertical direction are too large, so that there are only zero entries in the description of the sparse point cloud. This starts at 86.5 meters. From this distance, the (normalized) Euclidean distance corresponds to the Euclidean norm of the description of the reference pillar, so that a constant value can be seen in all graphs.

It can be concluded that the presented descriptors largely depend on the distance. This is due to the fact that the selected algorithms need a relatively large amount of points on an object to capture the geometric information. For example, a histogram-based description's resolution is only as accurate as the number of points in the local neighborhood, e.g., for four points 25%.

6. Spatial Allocation Accuracy:

In this experiment, an additional property of the descriptors is investigated, besides Definition 3.4. Here, the accuracy with which an object can be spatially detected within a typical scenery using the presented descriptors, FPFH, SHOT, and RoPS, is investigated. This is a good indicator to deduce the localization accuracy when using a specific object with the selected descriptors. The experiment is performed on the example of a vertical edge but could be carried out with other objects. Since usually no point hits the edge directly, the edge is a compelling case to study.

The descriptions of points on the same layer of the LiDAR sensor are compared to the reference description of an edge point. Again the medoid description is chosen as reference, using Equation 3.2, 3.3, and 3.4 on page 65. These distances are the input for the estimation of the standard deviation σ . The standard deviation indicates how precisely the localization algorithm can associate an edge. In this case, an inverse Gaussian normal distribution is assumed to calculate the standard deviation from this with a variation of the Full Width at Half Maximum (FWHM)² h :

$$\sigma = \frac{h}{2\sqrt{2 \ln(2)}}. \quad (3.5)$$

²The FWHM of a function with a maximum is the difference between the two argument values for which the function values have dropped to half the maximum.

Table 3.3: Standard deviations measuring the allocation accuracy of an edge point computed with Equation 3.5 for the descriptors FPFH, SHOT, and RoPS. For a synthetic edge, modeled with the Velodyne VLP 16 model, and for a real data edge, sampled by the Velodyne VLP 32-C, with two radii r_{\min} and r_{plus} .

		σ for r_{\min} [cm]	σ for r_{plus} [cm]
FPFH	synthetic edge	33	76
	real edge	9	16
SHOT	synthetic edge	15	22
	real edge	42	44
RoPS	synthetic edge	27 8 3 4 16	49 16 41 31 25
	real edge	27 82 32 44 16	49 16 40 31 25

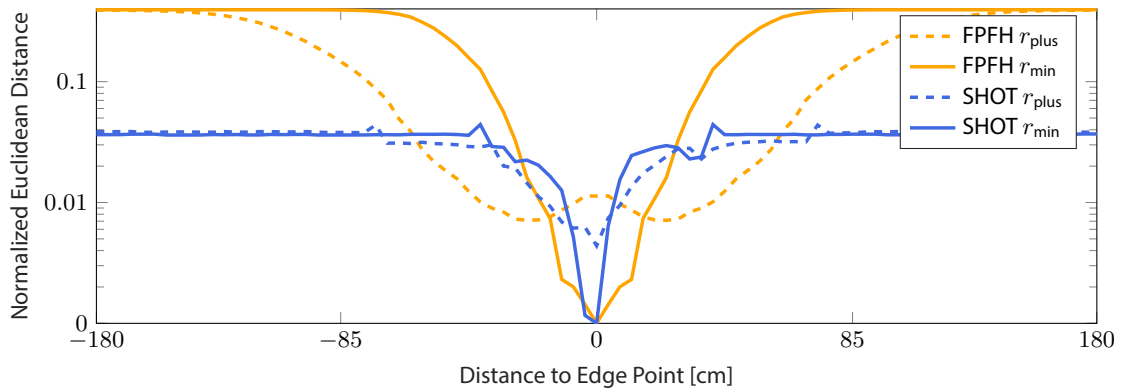
The standard deviation indicates how precisely an edge can be associated by the localization algorithm. The smaller the standard deviation is, the more the object is weighted in localization compared the other detected objects [160]. The standard deviation, therefore, has an important impact on the location accuracy.

In this experiment, two local neighborhood radii are used within the description calculation for each descriptor. One radius is the minimal local neighborhood radius r_{\min} for each descriptor, refer to Definition 3.5. That means the local neighborhood built with r_{\min} encloses three layers. The second radius is a radius that is nearly twice r_{\min} , called r_{plus} . This is chosen as this enables a local neighborhood enclosing five layers. Otherwise, the increase would evoke a small effect. The descriptions are computed for real and synthetic data of a building corner or an edge, respectively, the real data sampled on-board with the Velodyne VLP 32-C and the synthetic data modeled with the Velodyne VLP 16. The synthetic vertical edge is modeled by two planes perpendicular to each other. Figure 3.8 shows the evaluation with the synthetic data. Figure 3.9 illustrates the differences of the real data.

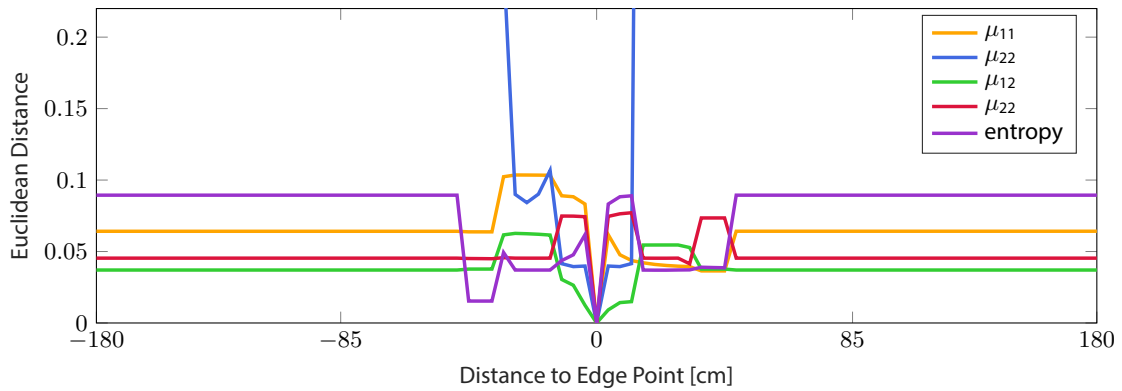
Both figures demonstrate that the closer a point is to the edge point, the more similar the descriptions get, except for discretization effects caused by the sparsely sampled point clouds. In the figures, the graphs show approximately a parabolic graph. This holds for every descriptor.

Looking at some sample standard deviations of Table 3.3 for synthetic data, the standard deviation computed with the FPFH descriptor and r_{\min} equals 33 centimeters. With r_{plus} it equals 76 centimeters. It can be deduced that the values of the standard deviation and the size of the local neighborhood radius correlate. The following applies: The larger the local neighborhood radius is chosen for edge points, the larger is the noise of the edge's normals, and the larger is the standard deviation. As a result, the edge is spatially less precisely assigned if the local neighborhood's radius size increases. The results show that the local neighborhood's radius is the upper limit for the size of the standard deviation. From this and the apparent differences between the object classes, it follows that, based on the descriptors presented, edges are particularly suitable for localization if the local neighborhood radius is chosen to be as small as possible in order to be still large enough, see Definition 3.5.

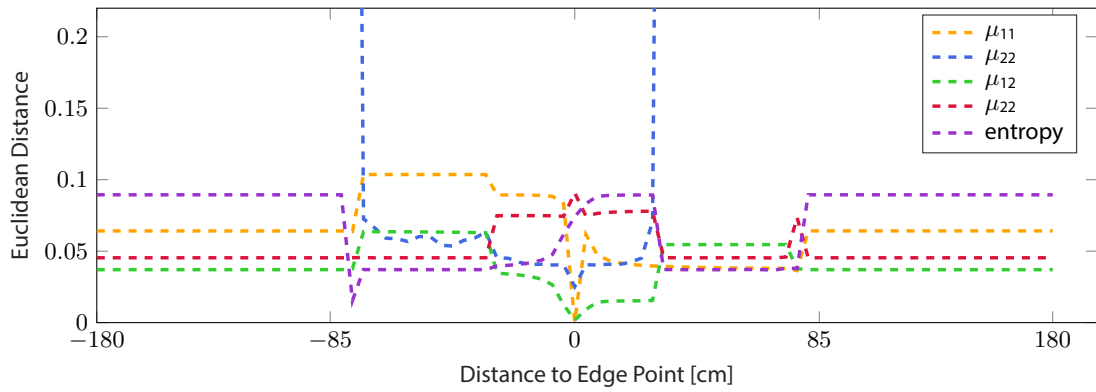
For the real data, the standard deviation determined with the FPFH descriptor and r_{\min} equals 9 centimeters. With r_{plus} it equals 16 centimeters. Also, for real data, the statements from before are true, i.e., that the local neighborhood size and the standard deviations correlate. The edge can be more precisely allocated compared to the results of the synthetic data. This is because a 32-layer LiDAR sensor was used in the real data experiment, which has a higher resolution than the LiDAR sensor model with 16 layers. Therefore, using the example of the edge and the differences of the object classes of previous experiments, it can also be shown in real data that the selected descriptors are suitable for localization in on-board data.



(a) FPFH and SHOT

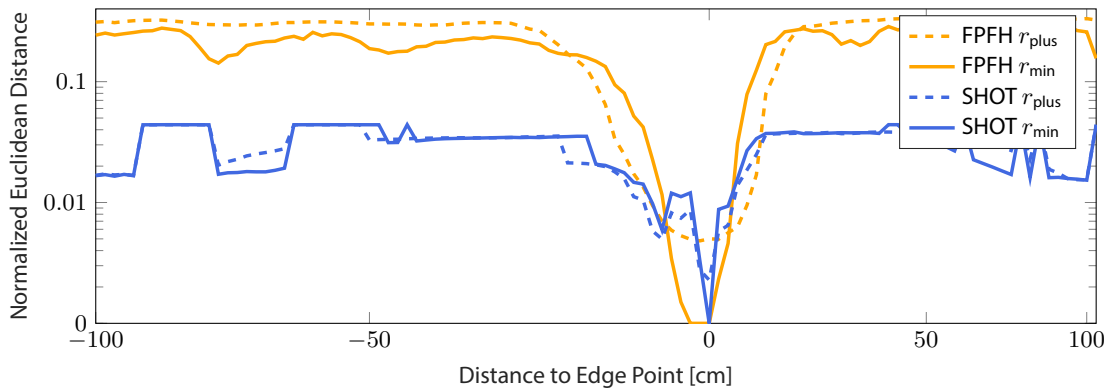


(b) RoPS with r_{\min}

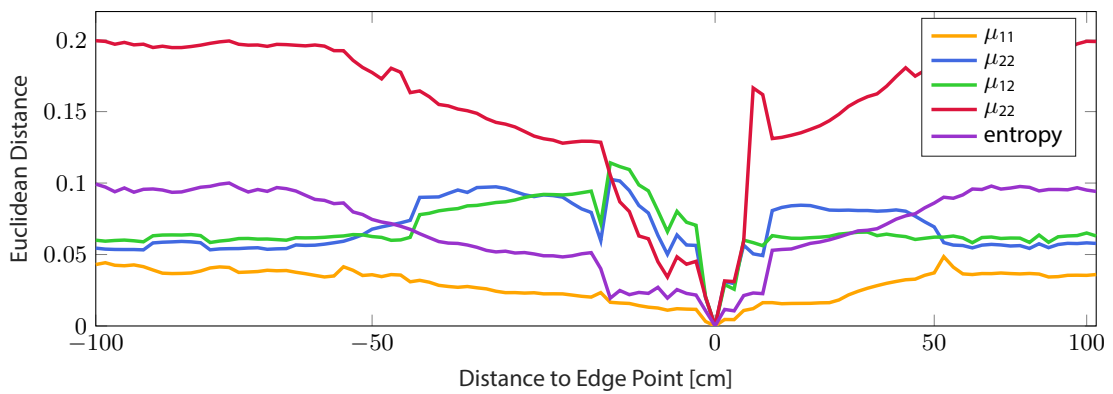


(c) RoPS with r_{plus}

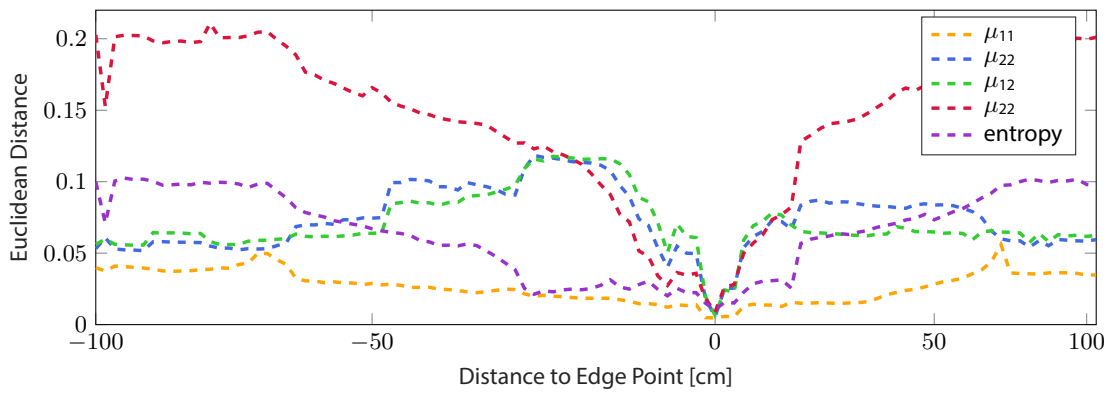
Figure 3.8: Description deviations of synthetic edge points, modeled after VLP 16, depending on the distance to the exact edge with the minimum local neighborhood radius r_{\min} and a larger radius r_{plus} according to Definition 3.4



(a) FPFH and SHOT



(b) RoPS with r_{\min}



(c) RoPS with r_{plus}

Figure 3.9: Description deviations of real edge points, sampled with VLP 32-C, depending on the distance to the exact edge with the minimum local neighborhood radius r_{\min} and a larger radius r_{plus} according to Definition 3.4

It can be summarized that the selected descriptors are able to spatially allocate objects both in synthetic and real data. The experiments have shown that the local neighborhood size correlates with the spatial allocation accuracy: the smaller the local neighborhood, the more precise the allocation. Thus, the local neighborhood should be chosen as small as possible if used for the localization task.

3.2.2. Own Developments and Extensions

In this subsection, the findings of the analyzes are reviewed to answer the first research question of this thesis. Next, it is derived in which respect the algorithms need to be improved to be useful in the context of *LiDAR-Feature-based Localization*. Subsequent follows the presentation of such improvements. These consist in definitions of functions determining the right choice of the local neighborhood, but also in novel descriptors.

Possibilities for Improvement

In the previous analyzes, it can be seen that the state-of-the-art descriptors meet several criteria of Definition 3.4, but not every criterion. Answering the first research question of this thesis: The geometry-based descriptors are suited in many aspects for a real-world localization, but some aspects need to be improved or compensated. In the following, the two main issues of the state-of-the-art descriptors are explained:

- The experiments show that the outcome of the descriptors largely depends on the local neighborhood size. The descriptions change with an increasing local neighborhood size until a characteristic description of an object is reached. For instance, if a local neighborhood is small, a description depicting the point cloud's curvature captures only planar characteristics. Therefore, an approach to determine the local neighborhood radius for a characteristic description of an object must be developed.
- Additionally, the analyzes show that the descriptors only work in short ranges, where the point cloud's density is large enough. The quality of the description decreases with increasing distance until no expressive description is possible. Based on the example of a pillar with a radius of 20 centimeters, this is reached at about 57.5 meters. For an automated vehicle, the localization requirements can be analyzed and from this, requirements on the specified range of descriptors can be derived. Reid et al. [104] examined the localization requirements, which have to be fulfilled for automated vehicles on local streets in the US. They say that the 1σ -accuracy of the heading should be 0.17° . Through the example of the pillar with a radius of 20 centimeters, the pillar must be detected at $\frac{0.2 \text{ m}}{\tan(\frac{0.17}{2} \cdot \pi / 180)} = 134.8$ meters for a 1σ -accuracy. This is more than twice the distance which can be reached with the state-of-the-art descriptors. However, it makes clear that the state-of-the-art descriptors do not reach the required independence on the distance. Thus, a novel descriptor should be developed to describe objects with a large distance to the sensor.

Determining Characteristic Local Neighborhood Size

A method for an automatic choice of the local neighborhood radius is proposed in the following, to ensure that the computed description is characteristic for each object. This has been become very clear in the experiment **1. Influences of the Local Neighborhood Size**. If the local neighborhood radius is too small, the description of a pillar looks like a planar object. If the local neighborhood radius is too large, further objects are enclosed by the local neighborhood and the description is no longer characteristic for the pillar. Therefore, the definition of the characteristic radius of a local neighborhood is introduced.

Definition 3.7 (Characteristic Radius of a Local Neighborhood). Let D be a descriptor function, let $\mathcal{N}_{\mathcal{P},r_1}, \dots, \mathcal{N}_{\mathcal{P},r_1+\varepsilon \cdot i}, \dots, \mathcal{N}_{\mathcal{P},r_1+(k-1) \cdot \varepsilon}, \dots, \mathcal{N}_{\mathcal{P},(r_1+(k-1) \cdot \varepsilon)+\varepsilon \cdot i}$ be $k \cdot i$ local neighborhoods with increasing radii $r_1 < r_1 + \varepsilon \cdot i < \dots < r_1 + (k-1) \cdot \varepsilon < (r_1 + (k-1) \cdot \varepsilon) + \varepsilon \cdot i$ around the same point \mathbf{p} with an appropriate radius increment $\varepsilon > 0$ and an appropriate number $i > 1$, let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be a descriptor-specific metric, let

$$\begin{aligned} e_{1,1} &= m_d(D(\mathcal{N}_{\mathcal{P},r_1}), D(\mathcal{N}_{\mathcal{P},r_1+\varepsilon})), \dots, \\ e_{1,i-1} &= m_d(D(\mathcal{N}_{\mathcal{P},r_1+\varepsilon \cdot (i-1)}), D(\mathcal{N}_{\mathcal{P},r_1+\varepsilon \cdot i})), \dots, \\ e_{k,1} &= m_d(D(\mathcal{N}_{\mathcal{P},r_1+(k-1) \cdot \varepsilon}), D(\mathcal{N}_{\mathcal{P},r_1+k \cdot \varepsilon})), \dots, \\ e_{k,i-1} &= m_d(D(\mathcal{N}_{\mathcal{P},(r_1+(k-1) \cdot \varepsilon)+\varepsilon \cdot (i-1)}), D(\mathcal{N}_{\mathcal{P},(r_1+(k-1) \cdot \varepsilon)+\varepsilon \cdot i})) \end{aligned}$$

be $k \cdot i$ differences between all descriptions of consecutive local neighborhoods, and let $\sigma_1 = \text{stdd}(e_{1,1}, \dots, e_{1,i-1}), \dots, \sigma_k = \text{stdd}(e_{k,1}, \dots, e_{k,i-1})$ be k standard deviations of description differences. A radius r_1 is called characteristic, if the standard deviations $\sigma_1, \dots, \sigma_k < \epsilon$ for an $\epsilon > 0$. If there is no smaller radius r_1 fulfilling this requirement, the radius is called minimal characteristic.

This definition further specifies the local neighborhood radius compared to Definition 3.5. It ensures that the three-dimensional structure of the point environment is characterized within the so defined local neighborhood, but also that the description captures the geometry of each object.

Here, two assumptions are made to determine the characteristic radius for each object automatically. First, it is assumed that the description values change marginally for radii greater or equal to the minimal characteristic radius. It is also assumed that the description values change smoothly. This assumption is made plausible by the previous experiment on the local neighborhood size.

In the following, the definition is explained in more detail. Considering a radius r_1 , i increments ε are added to r_1 . For every radius, all i descriptions are determined with a descriptor D . The differences $e_{1,1}, \dots, e_{1,i-1}$ are computed with a descriptor-specific metric m_d . The standard deviation σ_1 of the $i-1$ differences yields an information on the scattering of the description differences. For radii r_1, \dots, r_k , the standard deviations $\sigma_1, \dots, \sigma_k$ are computed in this way. If all standard deviations are smaller than a small enough threshold $\epsilon > 0$ the radius r_1 is characteristic. I. e. small scattering indicates that with increasing radii the descriptions only slightly differ from one another since the same object is always described by the descriptions. The smallest radius fulfilling this is defined as the minimal characteristic radius.

For that, the (normalized) Euclidean distances according to Equations 3.2, 3.3, and 3.4 are computed between consecutive descriptions of increasing local neighborhood radii. The increment of the radii must be chosen appropriately. For increasing local neighborhood radii around the same point, the minimal characteristic radius is reached, if several standard deviations of several consecutive description distances are smaller than a predefined threshold.

Additionally, an upper bound of the characteristic radius is determined to limit the radius, so that the neighborhood does not enclose other objects. It is assumed that the descriptions change a lot for radii greater or equal to the minimal characteristic radius, and the local neighborhood encloses other objects. The upper limit of the characteristic radius is chosen, if one standard deviation of several consecutive description distances is greater than a predefined threshold.

In this thesis, the experiments on several objects with all descriptors of a dense point cloud have shown that with an increment of one centimeter and a standard deviation, which is computed based on five consecutive description distances, smaller or greater than 0.01 yield a reasonable result of the characteristic radius.

Figure 3.10 illustrates the results of this calculation on a dense real-data wall on the example of the FPFH descriptor. The normalized Euclidean distance is displayed as a function of the local neighborhood size for each angle of the FPFH description. The minimal characteristic radius is marked as a red

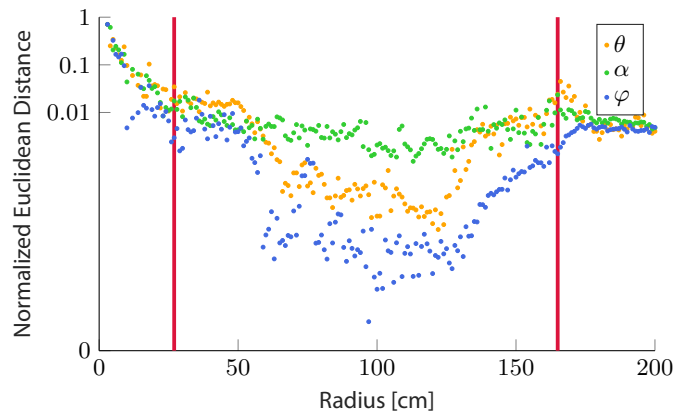


Figure 3.10: Normalized Euclidean distance between two descriptions computed with local neighborhoods with increasing radii as a function of the local neighborhood radius in a logarithmic scale for every angle of the FPFH description using the example of a dense real-data wall. The minimal characteristic radius and the upper limit of the characteristic radii are marked (red).

line at 27 centimeters. For small local neighborhood radii, the normalized Euclidean distances are large. This is since the local neighborhood radius is not large enough to yield expressive descriptions. Then, confirming the first assumption, the normalized Euclidean distances remain small because the local neighborhood encloses the same object points. The upper limit of the characteristic radii is marked at 165 centimeters, which corresponds to the distance from the center of the local neighborhood to another object, i.e., a building edge. As seen in further experiments, the upper limit can only be determined accurately if the local neighborhood encloses enough points of another object. Otherwise, the percentage of the other object's points is too small to change the description. This is no drawback, as the descriptions stay the same, i.e., the description still represents the object.

In Figure 3.11, the histograms of the FPFH descriptor are shown as a function of the local neighborhood size from the same point cloud as before. The three angles θ , α , and φ are presented separately. Each bin of the 11 bins is marked in the same color, while the 11 bins are stacked for every description. It can be seen that the minimal characteristic radius (at 27 centimeters) is marked, where the descriptions even out. The upper limit of the characteristic radii is set (at 165 centimeters), where the descriptions begin to change.

Long-Range Descriptors

In this part, two novel descriptors, together with some variations of them, are presented, which are capable of describing objects with a large distance to the sensor characteristically. Thus, they are called long-range descriptors.

First, the points of the input point cloud are selected whose local neighborhood provides sufficient information to calculate a description. Next, the local neighborhood's transformation around the query point into a viewing-angle independent LRF is presented. Afterward, the transformed local neighborhood's rasterization is depicted, which is the input for the description calculation. Then, every method of the actual description computation is presented, characterizing the neighborhoods' depth and point distribution information. Concluding, it is shown that the developed descriptors are able to depict objects with a large distance to the LiDAR sensor and able to differentiate between different objects.

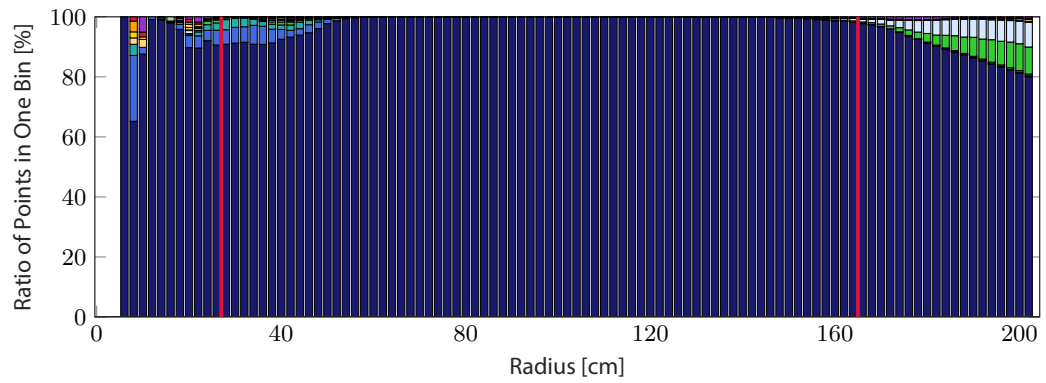
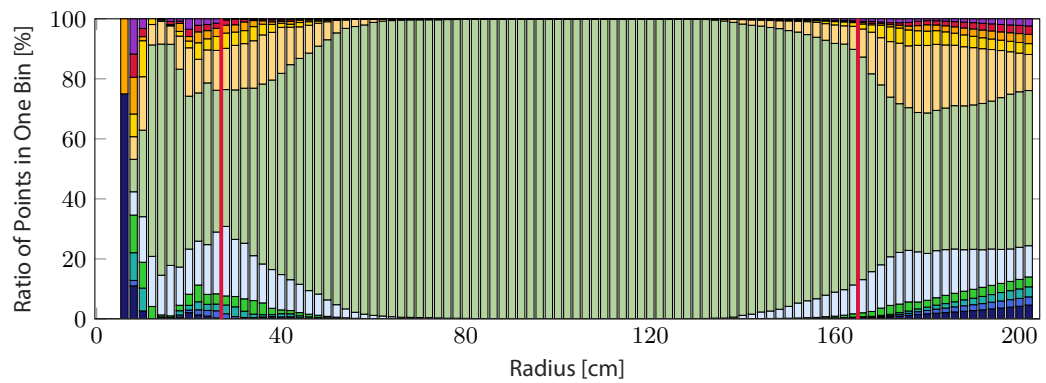
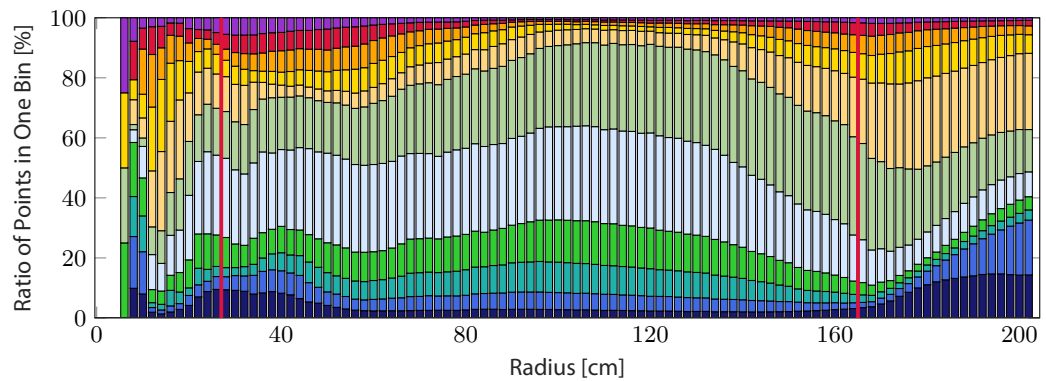
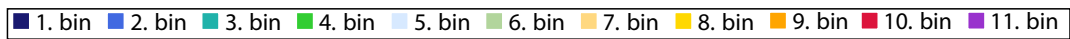
(a) First 11 bins of the FPFH description binning θ (b) Second 11 bins of the FPFH description binning α (c) Third 11 bins of the FPFH description binning φ 

Figure 3.11: The ratio of points in a histogram bin as a function of the local neighborhood radius for every angle of the FPFH description in a stacked bar using the example of a real data wall in a scene. The minimal characteristic radius and the upper limit of the characteristic radii are marked (red).

1. Preprocessing of the Point Cloud: First, those points of the point cloud are identified which are not suited for the description calculation due to lack of data or without value for the localization task. Those points are discarded whose local neighborhood radius is smaller than the minimum radius of Definition 3.5. In the practical implementation, the local neighborhood radius is smaller than the minimum radius, if one of the two following requirements are fulfilled. First, points on the outer layers of the point clouds are discarded since they only have one neighboring layer, and therefore a sufficient distribution of points within the local neighborhood is not present. Second, all points with a distance to the neighboring layer that is greater than half of the local neighborhood size are sorted out before the description calculation. If the distance is greater than that, the local neighborhood does not enclose this neighboring layer, which again leads to no sufficient distribution of points within the local neighborhood, refer to step 3 for more details. In both cases, the local neighborhood lacks sufficient environmental information in the local neighborhood so that misinterpretations can occur if only one layer is recorded, refer to Definition 3.5.

A disadvantage of this restriction is that half of the layers are no longer taken into account even at medium distances from the LiDAR sensor. However, increasing the overall size of the grid would, in turn, have the disadvantage of reduced descriptiveness.

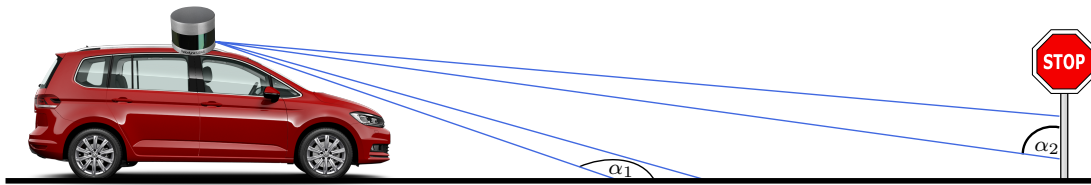


Figure 3.12: Illustration of two angles of incidence $\alpha_{1,2}$: The angle of incidence α_1 is larger than 100° , so that the associated sampling point is not taken into account for the description calculation. The angle of incidence α_2 is smaller than 100° , so that the associated sampling point is not taken into account for the description calculation.

Additional points are discarded for whom the angle of incidence of the LiDAR beam is greater than 100° since these points lie for the most part on the horizontal plane, e.g., in the street level, and have no use for the description calculation characterizing depth or point distribution information. Figure 3.12 visualizes the angle of incidence for a horizontal case and for a vertical case.

2. Transformation of the Local Neighborhood: The second step of the description calculation is the transformation of the local neighborhood points, which are measured in the VRF or SRF, into the LRF. If the point cloud is measured in the VRF, the sensors' calibration data is used for the transformation into the SRF. The transformation from the SRF into the LRF is necessary in order to achieve a viewing-angle independence. According to Definition 3.4 item (v), this step is crucial, since a description should be similar for the same object viewed from different angles. The transformation requires a calculation of a PCA, and thus the determination of the covariance matrix and eigenvalues and eigenvectors. This is performed based on Salti et al. [112]. The eigenvectors form the vectors of the LRF, the axes corresponding to the size of the eigenvalue. The x -axis is the eigenvector with the largest eigenvalue, the y -axis is the eigenvector with the second largest eigenvalue, and the z -axis is the eigenvector with the smallest eigenvalue. The z -axis is assumed the surface normal vector, because LiDAR sensors scan the surface of objects and the variance of the value is most likely the smallest one, which is orthogonal to the surface. The x - and the z -axis point in the direction where more points are located, the y -axis corresponding. Refer to Subsection 2.2.2 for a more detailed explanation of the PCA.

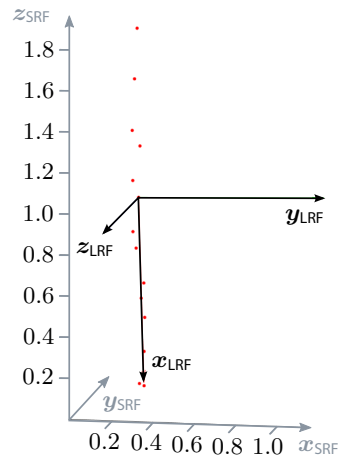


Figure 3.13: Transformation from the SRF (gray) into the LRF (black) on the example of a pillar, sampled by a Velodyne VLP 32-C

Figure 3.13 illustrates the definition of the LRF through the example of a pillar, which was sampled by a Velodyne VLP 32-C. The considered sample point whose local neighborhood is transformed is at the origin of the LRF. However, the determination of the LRF is optimized in that the covariance matrix is only calculated with a subset of the points from the local neighborhood. Downsampling is performed to compensate for point density differences within the local neighborhood. Differences in point density occur, for example, at building corners, on which one side is sampled less due to its viewing angle. This step stabilizes the LRF's viewing-angle independence.

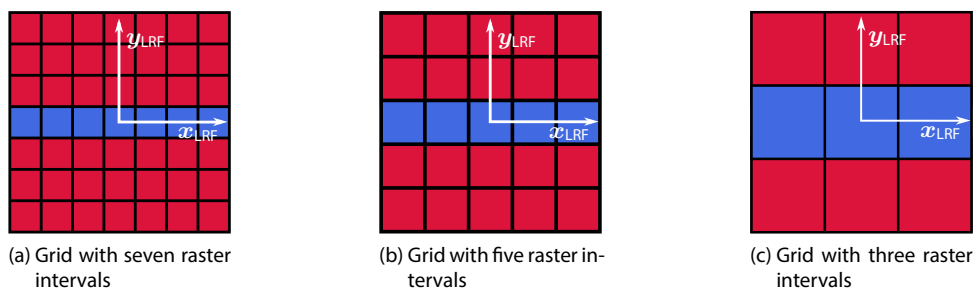


Figure 3.14: 2D Grid variants using the example of the pillar from Figure 3.13. The points' depth value (blue = low, red = high) on the pillar are rasterized in two dimensions in the LRF. The x_{LRF} -axis points in the direction of the greatest variance so that the pillar is rotated by 90° in the LRF.

3. Rasterization of the Local Neighborhood: After the local neighborhood has been transformed into a viewing-angle independent coordinate system, it is rasterized to determine the descriptions with the neighborhood's center in the grid's origin. Depending on the distance of the feature point to the LiDAR sensor, this is done in 2D or 3D grids with seven, five or three raster intervals along the two or

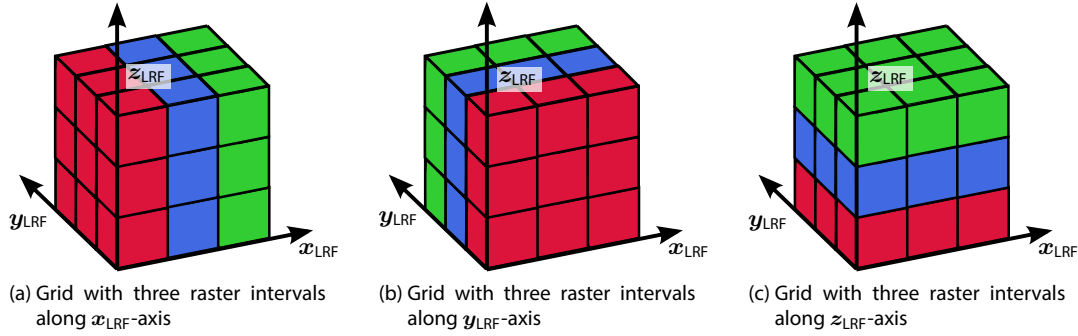


Figure 3.15: 3D Grid variants. The points distributions are rasterized in three dimensions along each axis of the LRF depending on their distance. In this example, each axis is divided into three parts. Each color indicates the considered cells.

three axes of the LRF. Rasterizing the local neighborhood is the elementary step in achieving distance-independent behavior since only one point per grid cell is required to produce a similar effect as with a high-resolution point cloud. The aim is to apply the different rasterizations at different distances from the LiDAR sensor and thus adapt the level of detail in the rasterization to the decreasing point density.

The variant with seven raster intervals has the highest descriptiveness and is used in the smallest distance range in which there is a sufficiently high point density. The variant with five raster intervals has a lower level of detail and is used to describe objects in medium distances. The variant with three raster intervals has the lowest descriptiveness and is used in the largest distance range since the point density is lowest in this area.

In the following, the size of the local neighborhood, the size of the cells of the three variants, and the three distance ranges are derived from the Velodyne VLP 32-C sensor's specification as this sensor has the highest resolution and thus still can achieve a reasonable density at a large distance. On the example of the pillar with a radius of 20 centimeters, refer to above, points of three layers fall onto the pillar of two meters height until 80 meters. After that, it cannot be secured that a pillar is sampled with at least three points. Thus, 80 meters are picked as an upper distance range limit for the description calculation. For the derivation, the vertical resolution $\alpha_{\text{vert}} = 0.333^\circ$ is used. This is a simplification considering the Velodyne VLP 32-C as this sensor has a non-uniform layer distribution, refer to Section 3.1. The simplification is chosen as the middle 16 layers of the Velodyne VLP 32-C have the highest vertical resolution of 0.333° and thus, at least 16 layers are reliably taken into account here [150].

The size of the local neighborhood r is determined with the largest distance of 80 meters, as the local neighborhood size should be the same at every distance but the resolution is lowest at 80 meters. A minimal grid cell size s_{min} is determined:

$$s_{\text{min}} = \tan(0.333^\circ \cdot \frac{\pi}{180}) \cdot 80 \text{ m} = 0.465 \text{ m}. \quad (3.6)$$

For a distance range up to 80 meters, the grid with three raster intervals is used. Therefore, the total size of the grid s_{total} can be determined from the minimal grid cell size s_{min} :

$$s'_{\text{total}} = 3 \cdot s_{\text{min}} = 3 \cdot 0.465 \text{ m} = 1.395 \text{ m}. \quad (3.7)$$

The required local neighborhood radius r can be determined from the size of the grid s_{total} :

$$r = \frac{s'_{\text{total}} \cdot \frac{1}{r}}{\cos\left(\frac{\pi}{4}\right)} = 0.986 \text{ m.} \quad (3.8)$$

Here, the radius is rounded to one. Rewriting Equation 3.8 with $r = 1$, yields:

$$s_{\text{total}} = \cos\left(\frac{\pi}{4}\right) \cdot 2 = 1.414 \text{ m.} \quad (3.9)$$

For the three variants of the grid cells, this yields the following grid cell sizes $s_{3,5,7}$:

$$\begin{aligned} s_3 &= \frac{s_{\text{total}}}{3} = \frac{1.414 \text{ m}}{3} = 0.471 \text{ m} \\ s_5 &= \frac{s_{\text{total}}}{5} = \frac{1.414 \text{ m}}{5} = 0.283 \text{ m} \\ s_7 &= \frac{s_{\text{total}}}{7} = \frac{1.414 \text{ m}}{7} = 0.202 \text{ m.} \end{aligned} \quad (3.10)$$

Following, the sizes of the individual grid cells are used to determine the upper limits of the distance ranges, in which the three different grids are valid:

$$\begin{aligned} d_3 &= \frac{s_3}{\tan \alpha_{\text{vert}}} = \frac{0.471 \text{ m}}{\tan\left(0.333^\circ \cdot \frac{\pi}{180}\right)} = 81.11 \text{ m} \\ d_5 &= \frac{s_5}{\tan \alpha_{\text{vert}}} = \frac{0.283 \text{ m}}{\tan\left(0.333^\circ \cdot \frac{\pi}{180}\right)} = 48.66 \text{ m} \\ d_7 &= \frac{s_7}{\tan \alpha_{\text{vert}}} = \frac{0.202 \text{ m}}{\tan\left(0.333^\circ \cdot \frac{\pi}{180}\right)} = 34.76 \text{ m.} \end{aligned} \quad (3.11)$$

Thus, the variant with seven raster intervals is used up to 30 meters, the variant with five raster intervals from 30 meters to 45 meters, and the variant with seven raster intervals from 45 meters to 80 meters. If the variant with seven or five raster intervals is used in larger distances with a lower point density, it is not ensured that points fall into the individual grid cells. This would distort the descriptions and should be prevented.

4. Description Calculation: After the local neighborhood has been transformed into a viewing angle independent coordinate system and the points have been rasterized into a grid to generate distance-independence, the descriptions are calculated. Therefore, two types of novel descriptors and some versions are introduced in the following: the Depth Leap Local (DeLL), a simplified version of it, and the modified Histogram of Point Distributions (HoPD) versions.

4.a Depth Leap Local Descriptor: The Depth Leap Local (DeLL) descriptor is based on the idea of the Daisy descriptor by Tola et al. [132], refer to Subsection 2.2.5. The main idea of the proposed descriptor is to characterize the 2D shape of the depth information of the aligned local neighborhood around a point using gradients. A variation of the Daisy descriptor is realized independently for every 2D grid variation, refer to Figure 3.14.

Initially, the grid cells $\mathbf{G}(j, k)$ at position (j, k) are filled with the mean relative distance of all M points $\mathbf{p} = [x_i, y_i, z_i]$ specified in the VRF or SRF falling within the grid cell:

$$\mathbf{G}(j, k) = \frac{1}{M} \cdot \left(\sum_{m=1}^M \sqrt{x_m^2 + y_m^2 + z_m^2} - \min_n \left(\sqrt{x_n^2 + y_n^2 + z_n^2} \right) \right). \quad (3.12)$$

Table 3.4: Daisy parameters of all three grid variants

parameter	grid w. 7 intervals	grid w. 5 intervals	grid w. 3 intervals
radius r	3	2	1
radius quantization q	3	2	1
angle quantization t	8	8	8
histogram quantization h	8	8	8
grid points s	25	17	9
size of description d	200	136	72

The grid is the input of the DeLL description calculation for the first step: determining the grid points of the Daisy descriptor. The radius r , radius quantization q , and angle quantization t define the number and positions of the grid points, refer to Subsection 2.2.5. While the angle quantization is identical for all three 2D grid variants with $t = 8$, the radius and the radius quantization change depending on the grid variant. For the variant with seven raster intervals, the radius is $r = 3$ and the radius quantization is $q = 3$. For the variant with five raster intervals, the radius is $r = 2$ and the radius quantization is $q = 2$. For the variant with seven raster intervals, the radius is $r = 1$ and the radius quantization is $q = 1$. Figure 3.16 shows the three grid variants and the grid points according to the Daisy descriptor. The radius and the radius quantization correspond to the number of grid cells. Thus, the description reproduces the information content adapted for each raster. The number of grid points s is calculated according to Equation 2.32, and the size of the description d is computed as $d_s = s \cdot h$, with histogram quantization $h = 8$ according to [132]. The resulting parameters are summarized in Table 3.4.

In the second step of the DeLL description calculation, the gradient images, i.e., orientation maps, are determined. This is done as described in Subsection 2.2.5 but is supplemented by an extension. In this case, the input image, one of the three grid variants filled with the depth information, is first smoothed and then convoluted in x - and y -directions with a filter operator to obtain the orientation maps. Both in the smoothing and the calculation of the orientation maps, border effects can occur. In order to avoid these border effects, an expansion of the input image is performed. For this purpose, the input grids are expanded by two grid cells in each direction.

After smoothing and calculating the orientation maps with the extended grids, the original sizes of the grids are used and the border grid cells are discarded. The h orientation maps are calculated analogously to the original Daisy descriptor, refer to Subsection 2.2.5. The q different standard deviations for the smoothing of the orientation maps are determined according to Equation 2.35 and are listed for the three different grid variants in Table 3.5.

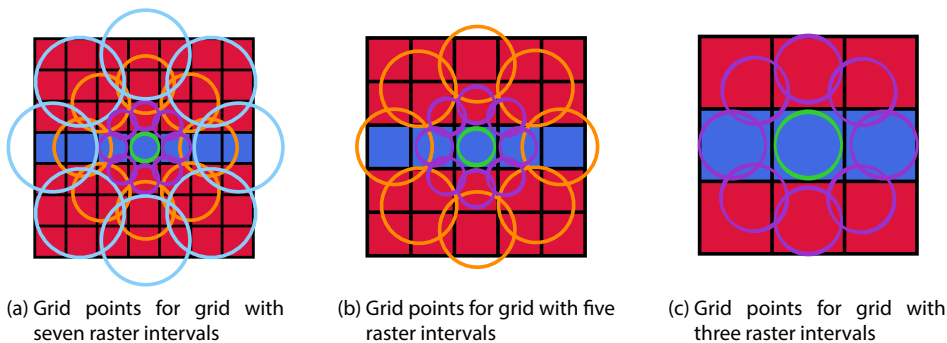


Figure 3.16: Grid points of the Daisy descriptor for each of the three grid variants

Table 3.5: Standard deviations for the smoothing of the orientation maps for every grid variant

standard deviation σ_i	grid w. 7 intervals	grid w. 5 intervals	grid w. 3 intervals
first layer: σ_1	1	1	1
second layer: σ_2	1.5	1.5	-
third layer: σ_3	2	-	-

The final step is the actual description calculation. This is done based on the smoothed orientation maps exactly as described in Subsection 2.2.5, but with an adapted normalization of the descriptions. While Tola et al. normalize each row to length one, here, the entire DeLL description D is normalized to D_{norm} with its largest gradient value:

$$D_{\text{norm}_{i,j}} = \frac{D_{i,j}}{\max_{m,n}(D_{m,n})} \forall i, j. \quad (3.13)$$

As a result, the relation of the descriptions is preserved. In the following, the largest gradient of the descriptor is referred to as the normalization factor and is stored in the descriptor as an additional entry.

Consequently, the variant with seven raster intervals results in $d = 200$ descriptions. The variant with five raster intervals results in $d = 136$ descriptions. And the variant with three raster intervals results in $d = 72$ descriptions. In each case, a further entry for the normalization factor is added.

4.b Simplified Depth Leap Local Descriptor: The simplified Depth Leap Local (DeLL) descriptor is a highly reduced version of the DeLL descriptor to reduce the description's size, albeit that this causes less distinctiveness compared to the DeLL. It is also based on the three different grid variants filled with depth information and applying the Daisy descriptor to it. The simplified DeLL descriptor is designed to reduce the DeLL description to its first row, thus, corresponding to the description entries of the central grid point. Accordingly, there are eight description entries, regardless of the underlying grid, which are normalized so that the largest gradient is equal to one. These description entries are relocated, maintaining their previous order, so that the largest value, which is equal to one after the normalization, is vector entry one. This creates a total viewing angle independence but neglects the spatial information of the description.

4.c Modified Histogram of Point Distribution: The main idea of the modified Histogram of Point Distributions (HoPD) is to describe the point distributions in 3D grid variations, refer to Figure 3.15. This is done by counting the points specified in the VRF or SRF in grid cells $\mathbf{G}(j, k, i)$ at position (j, k, i) of size $s_{3,5,7}$ around query point $\mathbf{p}_j^{x,y,z}$, refer to Equation 3.11. This number is normalized based on the total number of points within the local neighborhood of size s_{total} :

$$\mathbf{G}(j, k, i) = \frac{1}{|\{\mathbf{p}^{x,y,z} \in \mathcal{P}^{x,y,z} \mid \|\mathbf{p}^{x,y,z} - \mathbf{p}_j^{x,y,z}\|_{\infty} \leq s_{\text{total}}\}|} \cdot |\{\mathbf{p}^{x,y,z} \in \mathcal{P}^{x,y,z} \mid \|\mathbf{p}^{x,y,z} - \mathbf{p}_j^{x,y,z}\|_{\infty} \leq s_{3,5,7}\}| \quad (3.14)$$

The idea is based on the descriptor called 3DHoPD by Prakhya et al. [97]. While Prakhya et al. divide the intervals between the minimum and maximum coordinates of each spatial axis, in this thesis, these are divided like the three different 3D grid variants across the entire local neighborhood. Since the restrictions regarding point density apply here as well, the three different grid variants are only used in the associated distance range as defined before. Three versions of the modified HoPD are introduced in the following.

The simple version of the modified HoPD is called **uncorrelated** HoPD. For every grid variant, the number of points with the first, second, or third coordinate discretized is binned into a histogram. Therefore, each spatial axis is divided into cuboids, refer to the colored cuboids of Figure 3.15. This results in a 21-dimensional description for the variant with seven raster intervals, a 15-dimensional description for the variant with five raster intervals, and a nine-dimensional description for the variant with three raster intervals.

Another version of the modified HoPD is called **correlated** HoPD. The three spatial axes are considered together rather than individually. Therefore, the 3D local neighborhood is divided into equally sized cubes, refer to the cubes with black borders of Figure 3.15. Each cube's points are counted and divided by the total number of points in the local neighborhood. Each cube corresponds to one value in the correlated HoPD. This results in a 343-dimensional description for the variant with seven raster intervals, a 125-dimensional description for the variant with five raster intervals, and a 27-dimensional description for the variant with three raster intervals.

A third version of the modified HoPD is called **smoothed HoPD**. The input for the smoothed HoPD is the description of the uncorrelated HoPD. This description is smoothed to compensate for the non-uniform sampling of point clouds. The smoothing is done by filtering the uncorrelated HoPD with the smoothing operator $H = [\frac{1}{4}, \frac{1}{2}, \frac{1}{4}]$. External pixels are assumed to be zero. As the smoothing can lead to the sum of the descriptions of an axis no longer equal to one, normalization is carried out again after smoothing. Like the uncorrelated HoPD, the smoothed HoPD has 21, 15, or nine descriptions depending on the number of intervals per coordinate axis.

5. Analyzes: In the following, the descriptors are analyzed with two experiments. The first experiment determines distances of descriptions of a synthetic pillar computed with every of the developed descriptors to investigate their distance independence. In preceding experiments, it was shown that state-of-the-art descriptors lack this requirement. In the second experiment, distances of the descriptions of a real-world pillar and corner computed with every of the developed descriptors are compared. This is performed to examine whether the descriptor variants with seven intervals is more expressive than the variants with five or three intervals.

5.a Analysis on Dependence on Distance: In the following, it is demonstrated that the presented descriptors are capable of characteristically describing objects even with a large distance to the sensor.

For that, the same experiment as in **5. Dependence on Distance** analyzing the distance behavior is performed. Nevertheless, a short summary of the methodology of this experiment is given in the following.

A synthetically sampled pillar with a radius of 20 centimeters is used again as a sample object to examine the distance behavior, as it is known that this object can be stably detected applying semantic prior knowledge [14, 119, 123, 160]. In this case, the Velodyne VLP 32-C model is employed.

The descriptions d^1 of each distance of this sparse point cloud are compared to the description d^{ref} of a densely sampled pillar using a normalized Euclidean distance m_{normeucl} . For this, the descriptions of the point on the middle of each pillar are considered. The normalized Euclidean distance is computed as follows according to every descriptor, where d_n represents the n -th description entry:

$$\text{DeLL: } m_{\text{normeucl}}^{\text{DeLL}} = \frac{1}{\max_{\text{DeLL,ref}}} \cdot \sqrt{\sum_{n=1}^N (d_n^{\text{DeLL,1}} - d_n^{\text{DeLL,ref}})^2} \quad (3.15)$$

For the DeLL, the largest sum of the descriptions from all points of the reference point cloud $\max_{\text{DeLL,ref}}$ is used for the normalization, since the DeLL is not a histogram distribution, but normalization is necessary for comparability between the three grid variants. The variable N specifies the number of de-

descriptions and, depending on the grid variant, is 200, 136, or 72.

$$\text{simplified DeLL: } m_{n \text{ eucl}}^{\text{s. DeLL}} = \frac{1}{\max_{\text{s. DeLL,ref}}} \cdot \sqrt{\sum_{n=1}^{N=8} (\mathbf{d}_n^{\text{s. DeLL,1}} - \mathbf{d}_n^{\text{s. DeLL,ref}})^2} \quad (3.16)$$

For the simplified DeLL, the Euclidean distance is normalized analogously to the DeLL by the largest sum of the descriptions from all points of the reference point cloud $\max_{\text{s. DeLL,ref}}$.

$$\text{uncorrelated HoPD: } m_{n \text{ eucl}}^{\text{uncorr. HoPD}} = \frac{1}{\sqrt{2} \cdot 3} \cdot \sqrt{\sum_{n=1}^N (\mathbf{d}_n^{\text{uncorr. HoPD,1}} - \mathbf{d}_n^{\text{uncorr. HoPD,ref}})^2} \quad (3.17)$$

$$\text{correlated HoPD: } m_{n \text{ eucl}}^{\text{corr. HoPD}} = \frac{1}{\sqrt{2}} \cdot \sqrt{\sum_{n=1}^N (\mathbf{d}_n^{\text{corr. HoPD,1}} - \mathbf{d}_n^{\text{corr. HoPD,ref}})^2} \quad (3.18)$$

$$\text{smoothed HoPD: } m_{n \text{ eucl}}^{\text{s. HoPD}} = \frac{1}{\sqrt{2} \cdot 3} \cdot \sqrt{\sum_{n=1}^N (\mathbf{d}_n^{\text{s. HoPD,1}} - \mathbf{d}_n^{\text{s. HoPD,ref}})^2} \quad (3.19)$$

The normalization of the HoPD variants is straight forward, as they are histogram-based descriptors.

Figure 3.17 shows the descriptors' distance dependence. First, several general statements about the three different grid variations are made, regardless of the descriptor algorithm.

The descriptor variants with grid variations of seven raster intervals are not robust at large distances, which is true for every descriptor algorithm. The descriptor variants with grid variation of three raster intervals are able to calculate equal descriptions up to 100 meters. The descriptor variants with grid variation of five raster intervals represents a trade off between the other two variations. However, the distance independence might counteract with the expressiveness of each variation, see the next experiment.

Looking at the description distances of the DeLL description, it can be seen that the description deviations are comparably small. The descriptions calculated with the grid with three raster intervals do not differ from the descriptions of the reference pillar. This is because points of the sparsely sampled pillar fall at every distance in the same raster as points of the dense pillar, i.e., because of the rough rasterization and the large grid cells. Therefore, the descriptions are the same. However, the descriptions computed with the grid with seven or five intervals differ at larger distances from the dense pillar. But this occurs mostly after their previously defined application range except of some discretization effects. Therefore, the effect is negligible. These discretization effects can be seen in some outliers and are due to the fine rasterization, whenever points of the pillar fall into neighboring cells due to the non-uniform resolution of the Velodyne VLP 32-C.

The graphs of the simplified DeLL look very similar to the one of the DeLL descriptor. The descriptions computed with the grid with three raster intervals are equal to those of the reference pillar. For the other two variants, the normalized Euclidean distances are larger compared to the DeLL descriptor. This is due to the fact that the simplified DeLL is based on the depth information of the grid's central point hardly considering the neighboring grid cells. Therefore, compared to the DeLL, it does not even out discretization effects by considering more cells if, e.g., no point falls into the central points.

The graphs of the uncorrelated HoPD show relatively large deviations compared to the other descriptors. If the pillar diameter is smaller than the grid cell size so that differences along the local x -axis only cause derivations of the normalized Euclidean distances, an association to the reference pillar is possible. However, if the points of the pillar fall into several sections in all three spatial directions, discretiza-

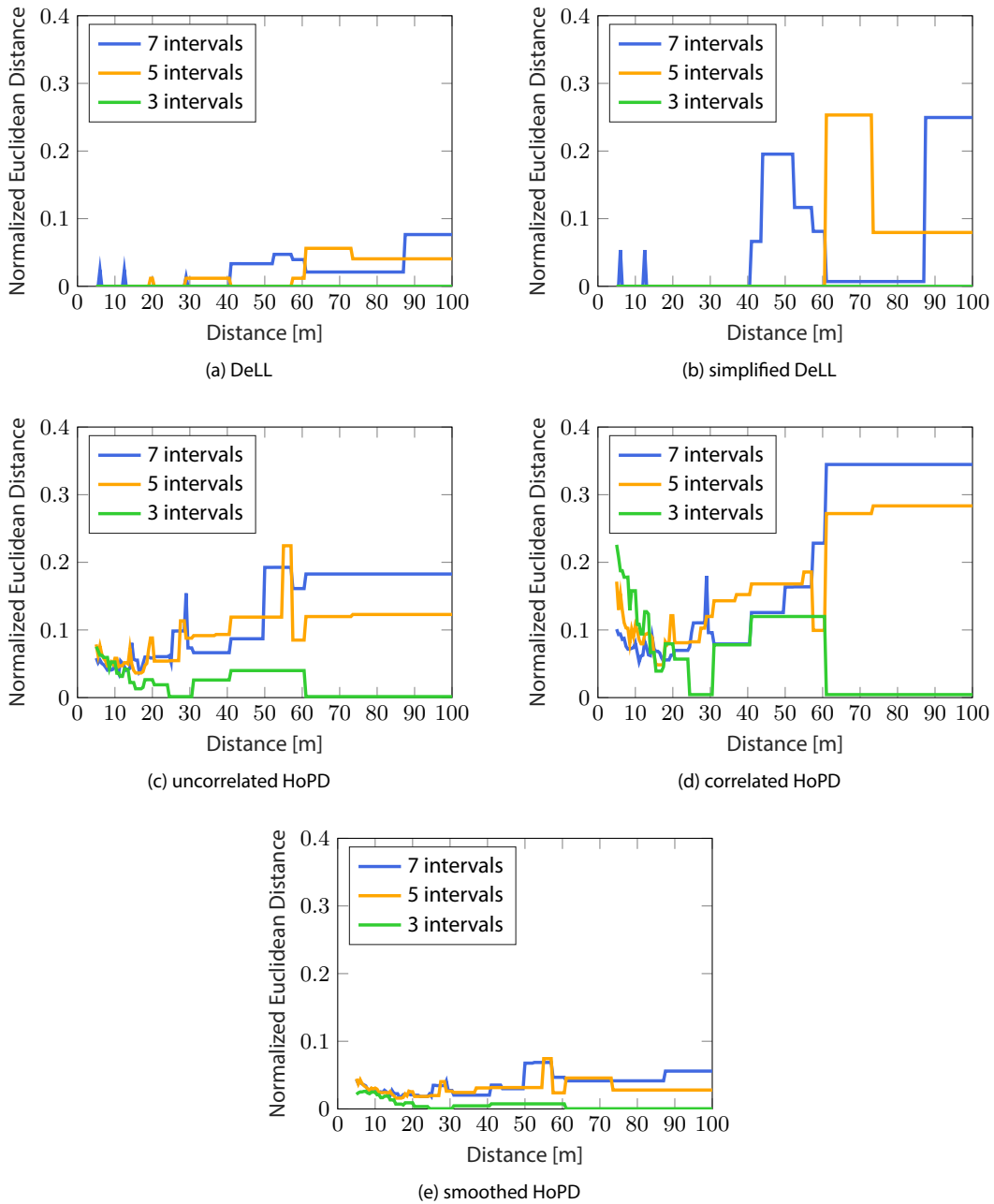


Figure 3.17: Description deviations of a sparse, synthetic pillar compared to a dense, synthetic pillar as a function of the distance to the pillar for the descriptors developed in this thesis

tion effects can be seen. Then, the descriptor fails with increasing distance, since occurring gaps in the grid in three spatial axes cause a large normalized Euclidean distance to the high-resolution reference. In summary, it can be stated that the distance behavior of the uncorrelated HoPD is not sufficient to meet the requirement for a description independent of the point density or distance.

In the case of the correlated HoPD, it can be seen, for the same reasons as for the uncorrelated HoPD, the correlated HoPD cannot describe the local neighborhood independently of the point density or distance. The uneven vertical scanning of the Velodyne VLP-32C and resultant gaps in the grid have an effect on all three axes of the LRF. By describing all three spatial axes in one histogram, the effect is very large and causes large degradation with larger distances. Thus, the normalized Euclidean distances computed with the correlated HoPD are larger than those determined with the uncorrelated HoPD.

The smoothed HoPD variant changes the least with increasing distance compared to the other HoPD versions and compared to the DeLL, because it compensates the gap problem of the other HoPD versions by smoothing the grid entries. Nevertheless, as the distance increases, there are occasionally stronger increases in the normalized Euclidean distance or outliers for the grid variant with seven or five raster intervals. However, these effects are relatively small and can be neglected. One additional effect has to be pointed out. Against expectations, up to 15 meters, the normalized Euclidean distance decreases. This can be explained looking at the different densities between the compared pillars. The number of points falling into the subdivisions differs from the high-resolution reference pillar which results in the small description differences. However, the smoothed HoPD shows the smallest deviations and is thus preferable to the other descriptors.

Compared to distance behavior of state-of-the-art algorithms, cf. Figure 3.7, it can be seen that the DeLL and the smoothed HoPD descriptor show smaller description deviations and thus outperform these algorithms. Therefore, the DeLL and the smoothed HoPD algorithms outperform that of the state-of-the-art and show a practicability for an application in localization.

5.b Analysis on Distinguishing Between Objects: The following experiment aims to investigate how descriptive the individual descriptors, i.e., DeLL, simplified DeLL, uncorrelated HoPD, correlated HoPD, and smoothed correlated HoPD, are depending on their rasterization variant. It is to be expected that the descriptor variant with seven intervals depicts the points of a local neighborhood more expressive than the variant with five or three intervals for each developed descriptor.

For this experiment, the point clouds of a real-world pillar and house corner sampled with the Velodyne VLP 32-C sensor are considered. A pillar, a house corner, and a wall are often objects that are used for localization, refer to previous experiments. However, the grid of a wall does not differ for grids with different number of cells. Therefore, only the pillar and the house corner are examined here. Both objects were sampled at approximately 8 meters in nearly the same conditions, i.e., weather, temperature, and time of day. The descriptions are determined of a point which is located on the middle of each object so that their descriptions capture the geometry of the object. Then, the (normalized) Euclidean distances between the two descriptions are computed for every descriptor. For the uncorrelated HoPD, correlated HoPD, and smoothed correlated HoPD descriptors, the normalized Euclidean distances according to Equations 3.17, 3.18, 3.19 are computed. For the DeLL and simplified DeLL descriptors, the Euclidean distances are calculated omitting the normalization factor of Equations 3.15 and 3.16 as these were the maximal distance, which does not make sense here. The following idea is assumed for the Euclidean distances: the higher the distance between the two objects, the more expressive the descriptor variant is able to characterize the geometry of the object.

In Table 3.6, the results of this analysis are listed. It can be seen for every descriptor that the more cells are used for the rasterization, the greater is the distance between the two objects. That means that the descriptor is more expressive when more cells are used. Thus, it makes sense to apply only the grid variant with three intervals, showing the best distance behavior, but to apply the grid variant to its specified distance to ensure descriptiveness.

Table 3.6: (Normalized) Euclidean distances between real-data corner and pillar for the descriptors DeLL, simplified DeLL, uncorrelated HoPD, correlated HoPD, and smoothed correlated HoPD

grid variant	DeLL	simplified DeLL	uncorrelated HoPD	correlated HoPD	smoothed HoPD
w. 7 intervals	0.024	0.0012	0.7	0.11	0.09
w. 5 intervals	0.027	0.0018	0.74	0.14	0.1
w. 3 intervals	0.029	0.0029	0.8	0.18	0.13

In summary, the developed descriptors create distance independence by rasterizing the local neighborhood. The analyzes on them demonstrate that the rasterizing can successfully solve the problem when suitably processing the rastered data. Here, the descriptors smoothing the 2D or 3D grid, i.e., DeLL and smoothed HoPD, show the best distance behavior and thus, overcoming the issue of the non-uniform sampling, which is essential when rasterizing. Concerning the normalized Euclidean distance at distances of five to 15 meters, the gradient-based descriptors DeLL and simplified DeLL have the advantage that the relative number of points per grid cell does not influence the description compared to the smoothed HoPD. However, the less cells are used the more is the descriptor distance independent. Therefore, a second experiment shows that it makes sense to apply a grid with more cells to ensure a descriptiveness of the descriptors.

3.3. Intensity-Based Descriptors

In this section, the state-of-the-art intensity-based descriptors are theoretically examined, and a novel descriptor, overcoming their issues, is introduced. This section answers the second research question of this chapter, whether also the intensity-based descriptors not-only the geometry-based are well-suited for localization.

3.3.1. Theoretical Analyzes on State-of-the-Art Algorithms

Intensity-based descriptors have received some attention in recent years.

Cop et al. [20] introduce the DDescriptor of LiDAR Intensities as a Group of HisTograms (DELIGHT) descriptor. It is designed as a global descriptor computing one vector for the whole point cloud. The description is calculated by dividing the point cloud into 16 parts and counting the differences of raw reflectivity returns in these subdivisions. A global descriptor is not suitable in the context of localization, as it is not very robust. Changes in parts of the environment affect the global description representing a scene, which may cause a large inaccuracy of localization with global descriptors. Additionally, only stable elements of the environment should be used for localization, which is hard to realize with global descriptors.

Guo et al. [42] refined the idea of the DELIGHT descriptor. Their algorithm, called Intensity Signatures of Histograms of Orientations (ISHOT) descriptor, is designed as a local descriptor that depicts a local neighborhood around a point of the point cloud instead of the whole point cloud at once. The ISHOT descriptor divides a spherical local neighborhood into multiple subdivisions, as introduced in the SHOT descriptor, refer to 2.4a in Subsection 2.1.1. Instead of counting the differences of intensity returns within each subdivision, the ISHOT method bins the intensity return in each part of the local neighborhood into a histogram.

Without any preprocessing step, both descriptors are dependent on different distances, viewing angles, and point cloud densities. They follow a simple approach by binning occurrences and differences of intensities into histograms. These approaches directly process the raw intensity data without adding a step to make the descriptors robust. They do not solve the problem of differently sampled

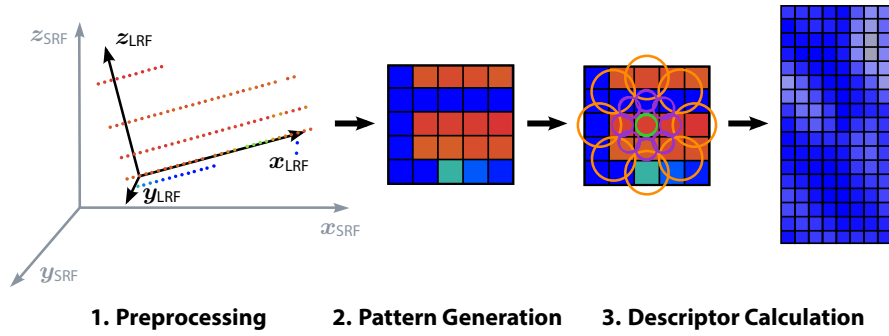


Figure 3.18: Scheme of the proposed GRAIL descriptor. 1. Points of a local neighborhood are normal aligned to be independent of the viewing angle, and 2. their intensities (blue = low, red = high) are binned into a grid. 3. With the Daisy description computation using the grid, the normalized description of the local neighborhood is calculated.

objects, like sampling the same object from different distances or non-uniformly sampled LiDAR point clouds. Accordingly, a novel descriptor based on intensity data overcoming these issues is introduced.

3.3.2. Developments

In the following, the intensity-based descriptor, developed in this thesis, is presented. The main idea of the descriptor, called GRADIENTS of Intensities as a Local descriptor (GRAIL), is based on the Daisy descriptor by Tola et al. [132], refer to Subsection 2.2.5, describing the shape of the intensity in the local neighborhood with gradients.

The computation of the description consists of three central steps, refer to Figure 3.18 for an illustration. At first, a spherical local neighborhood around every query point of the point cloud in the Sensor Reference Frame (SRF) is computed and transformed into a viewing independent Local Reference Frame (LRF). Next, the local neighborhood's intensities are rasterized, which is the input for a variation of the Daisy description calculation. The resulting description is normalized hereafter. The three steps are explained in more detail in the following.

1. Preprocessing: First, points of the point cloud are chosen which provide sufficient information to calculate a description. This procedure follows the idea of the long-range descriptors DeLL and HoPD versions, refer to Subsection 3.2.2. Thus, points are discarded whose local neighborhood radius does not fulfill Definition 3.5. That means that points on the outer layers of the point clouds are rejected, as only one neighboring layer exists. Including those points would falsify the descriptions as the shape of the intensities always forms an intensity edge. Furthermore, points with a distance to the neighboring layer that is greater than half of the total grid size are discarded. In this case, the local neighborhood does not enclose this neighboring layer, which also results in a misinterpretations if only one layer is included.

For each of the remaining points, the description is calculated using the local neighborhood. After the discarding process, the local neighborhood measured in the VRF or SRF is transformed into a viewing-angle independent LRF. As defined in Requirement (v) of Definition 3.4, a description should not change much if the same object is viewed from different angles. Hence, each spherical local neighborhood is transformed into a LRF obtained by a PCA, as recommended by Tombari et al. [134] or Salti et al. [112]. The eigenvectors of the PCA build the axes of the LRF, where the z -axis corresponds to the eigenvector with the smallest eigenvalue. Refer to Subsection 2.2.2 for an explanation of the PCA.

The z -axis is assigned as the surface normal vector. As LiDAR sensors scan the surface of objects, the variance of the value will most likely be the smallest, which is orthogonal to the surface. Thus, the vector with the smallest eigenvalue is assumed as the surface normal. Figure 3.13 and Figure 3.18 show examples of local neighborhoods being transformed from the SRF into the LRF.

2. Pattern Generation: Next, the transformed local neighborhood is rasterized into a 2D grid consisting of equally sized cells. This step is fundamental for enabling distance independence of the descriptor because at least one point in a grid cell is necessary, creating a similar description as with high-resolution point clouds. Here, the dimension along the normal axis is eliminated. In this way, the least information is lost during the reduction in two dimensions. Each equally sized square grid cell of a square grid $\mathbf{G}(j, k)$ at position j, k is assigned the maximum value of the points' reflectivities r_i falling into that grid cell:

$$\mathbf{G}(j, k) = \max_i (r_i). \quad (3.20)$$

If no point falls into a pattern's cell, the cell's value is set to zero.

Since the query point is located at the origin of the LRF, the query point falls in the middle grid cell. Consequently, the number of grid cells has to be odd. As introduced for the long-range descriptors, three different rasterizations at three different distance ranges, maximal up to 80 meters, are applied to adapt the level of detail in the rasterization to the decreasing density within a point cloud: with seven, five, or three raster intervals.

The grid with seven raster intervals is able to describe the most details of the intensity shape but requires the highest point density. Thus, it is used in the distance range near the sensor, i.e., zero to 30 meters, refer to Equation 3.11. The size of each grid cell is the smallest compared to the other two variants, i.e., 0.202 meters, refer to Equation 3.10. The grid with five raster intervals is able to describe fewer details of the intensity shape and is used at medium distances, i.e., 30 to 45 meters, refer to Equation 3.11. The size of each grid cell is 0.283 meters, refer to Equation 3.10. The grid with three raster intervals can describe the lowest level of details but can deal with the lowest point density. Therefore, it can be applied for points of the point cloud, which are far away, i.e., from 45 to 80 meters, refer to Equation 3.11. The size of each grid cell is 0.471 meters, refer to Equation 3.10.

3. Description Calculation: The 2D intensity-grids of the aligned local neighborhood around the query point are used to describe the intensities' shape using gradients. For this purpose, the Daisy descriptor originally introduced by Tola et al. [132], refer to Subsection 2.2.5, is adapted. A modification of the Daisy descriptor is applied separately for every grid variant.

The first step of the GRAIL descriptor is to determine the positions of the grid points of the Daisy descriptor for each cell. This is realized according to the determination of the parameters of the DeLL descriptor, refer to Subsection 3.2.2. The grid points' positions are defined by the radius r , radius quantization q , and angle quantization t . A visualization of the resulting grid points can be seen in Figure 3.16, and the resulting parameters are summarized in Table 3.4.

The second step of the GRAIL descriptor is to compute the gradient images, i.e., orientation maps. The orientation maps are computed as described in Subsection 2.2.5. Since the intensity grids are much smaller than the camera images used by Tola et al., the border effects need to be compensated. Initially, the intensity grids are smoothed. Next, they are folded in x - and y -directions with a filter operator in order to obtain the orientation maps. Both, in the smoothing and gradient calculation, values outside of the local neighborhood are consulted, i.e., the input images are expanded by two pixels in each direction. The values of the neighboring points are inserted into the additional pixels. Afterward, the border grid cells are discarded to regain the original size of the input grid. This grid is used to determine h orientation maps analogously to the original Daisy descriptor, refer to Subsection 2.2.5.

According to Equation 2.35, the q different standard deviations for the smoothing of the orientation maps are computed and are presented in Table 3.5 for each grid variant.

The third step of the GRAIL descriptor is the description calculation. The calculation is performed using the smoothed orientation maps, as depicted in Subsection 2.2.5. The normalization of the descriptions is executed differently from the one in [132] by Tola et al., where each row of the description is normalized to length one. In the case of the GRAIL descriptor, the whole description \mathbf{D} is normalized to \mathbf{D}_{norm} with its largest gradient value, refer to Equation 3.13. As a result, the relations of the intensity gradients are kept. From now on, the description's largest gradient is called the normalization factor and is saved in the description as an additional entry.

In summary, the calculations yield a description with $d = 200$ values in case of the grid with seven raster intervals, a description with $d = 136$ values in case of the grid with five raster intervals, and a description with $d = 72$ values in case of the grid with three raster intervals, in each case with an additional flag for the grid type and the normalization factor.

3.3.3. Analysis

In the following analysis, the suitability of the newly developed intensity-based GRAIL descriptor is compared to the selected geometry-based descriptors, i.e., FPFH, SHOT, and RoPS descriptors, by computing the sensitivity to changes of the relative movements of a test vehicle in real-world data.

Here, the accuracy of the approaches is examined by calculating the vehicle's odometry using consecutive sparse point clouds measured with the Velodyne VLP 32-C. In total 3592 LiDAR scans with a mean velocity of 8.79 m/s are considered. The scans were recorded in suburbs, highways and industrial areas during fair and rainy weather. The computed odometry is compared to the reference odometry, determined by the reference trajectory recorded with a high-precision Applanix POS LV 520 system [1], refer to Figure 3.1a for the mounting of the reference system in the test vehicle.

The descriptions are calculated for the preprocessed points within each point cloud, to determine the odometry based on the descriptions. The descriptions, together with the 3D information of the point cloud, are the input for an odometry calculation, which is realized with the Iterative Closest Point (ICP) [9], refer to Subsection 2.2.3. The ICP algorithm is an easy-to-implement method that serves as a sufficient indicator for the descriptor's suitability in the localization context. In this case, the ICP is applied to register two consecutive LiDAR point clouds and thus to determine the translation vector and rotation matrix between these consecutive point clouds based on the spatial and description information. As metric for comparing the spatial information the Euclidean distance is applied. As metric for comparing the description information Equations 3.2, 3.3, and 3.4 are used for the FPFH, SHOT, and RoPS descriptor. For the GRAIL descriptor, the following equation computes the differences between two descriptions $\mathbf{d}^{\text{GRAIL},1}$ and $\mathbf{d}^{\text{GRAIL},2}$:

$$m_{n \text{ eucl}}^{\text{GRAIL}} = \frac{1}{\max_{\text{GRAIL}}} \cdot \sqrt{\sum_{n=1}^N (\mathbf{d}_n^{\text{GRAIL},1} - \mathbf{d}_n^{\text{GRAIL},2})^2}, \quad (3.21)$$

where \max_{GRAIL} is the maximal difference occurring in the applied data set.

The ICP is applied with the rejection parameter option not considering 40% of the point pairs between the two consecutive point clouds. The rejection parameter of 40% implies that the point pairs with a distance that is greater than 60% of all distances are not taken into account when determining the translation vector and rotation matrix. In this experimental evaluation, it could be observed that 40% have yield the best odometry accuracies for all descriptors. For more details about the ICP and rejection parameter, refer to Subsection 2.2.3. The results yield the three-dimensional movement of the test vehicle between two point clouds.

Table 3.7: Average and standard deviation of absolute and rotational odometry errors calculated using the FPFH, SHOT, RoPS, and GRAIL descriptor for the collected data using Velodyne VLP-32C

descriptor	absolute [m]	rotational [rad]
FPFH	0.43 ± 0.1	0.23 ± 0.09
SHOT	0.5 ± 0.13	0.24 ± 0.09
RoPS	0.12 ± 0.13	0.24 ± 0.09
GRAIL	0.45 ± 0.19	0.2 ± 0.13

Table 3.8: Average and standard deviation of absolute and rotational odometry errors calculated using the GRAIL descriptor for the collected data using Velodyne VLP-32C comparing three different sceneries

sceneries	absolute [m]	rotational [rad]
highway	0.77 ± 0.11	0.19 ± 0.14
suburb	0.12 ± 0.16	0.15 ± 0.11
industrial	0.45 ± 0.29	0.27 ± 0.13

Table 3.7 reports the average absolute and rotational errors of the odometry as Root Mean Square (RMS). The absolute error is specified as the root-mean-square error and the rotational error as the radial angular difference between the two consecutive 3D driving directions. The results illustrate that the intensity-based descriptor, GRAIL, achieves similar odometry deviations compared to the other geometry-based descriptors, FPFH, SHOT, and RoPS. However, it can be seen that the standard deviation is much higher compared to the other descriptors. This is due to the fact that the position accuracy largely depends on the variation of the intensity data. If no high reflecting objects, like road signs, occur in a scene, the odometry accuracy decreases.

In Table 3.8, the average RMS absolute and rotational errors of the odometry of the three different sceneries are compared. It turns out that the results of the descriptor are highly scene dependent. On the one hand, the velocity has an influence on the odometry accuracy. The higher the velocity, the greater the inaccuracies of the odometry calculation. While the point clouds of the highway scenery are recorded with an average velocity of 19.5m/s, that of the industrial area scenery with an average velocity of 5 m/s, and that of the suburban are scenery with an average velocity of 3 m/s. The higher the velocity of the test drive, the greater the corresponding deviations from the reference odometry. A scan is recorded every 1/10 seconds, which means that at an average speed of 19.5 m/s for the highway scenery, 1.9 meter are traveled.

Summarizing, it can be said that the odometry accuracy of intensity-based descriptors are similar to those of geometry-based descriptors. However, their results are not as robust compared to geometry-based algorithms.

3.4. Summary

Based on the findings of the literature review provided in the previous chapter, this chapter covers the topic of descriptors for real-world data in the context of the *LiDAR-Feature-based Localization*. A description can be assigned to each point of the point cloud and captures numerically characteristic relations between sampling points in a local neighborhood around that point of a LiDAR point cloud.

At the core of this chapter is the definition of criteria for well-suited descriptors for automated vehicles' self-localization. They state that descriptors need to detect the local neighborhood structure expressively, that they need to distinguish between different object types, but characterize similar objects in the same way independent of the point cloud densities. Additionally, they need to be independent of the viewing angle and the distance between the LiDAR sensor and the object.

Following these criteria, the state-of-the-art descriptors characterizing the geometric information are analyzed thoroughly with real-world data or synthetic data modeling the real-world environment based on existing LiDAR sensors. In doing so, the first research question addressing the suitability of geometry-based descriptors for real-world localization is answered. These analyzes show that the state-of-the-art descriptors fulfill many of the criteria but lack the independence on the distance and depend largely on the size of the local neighborhood.

Therefore, some algorithms and modifications overcoming these issues are introduced in this chapter. This includes a method choosing the best local neighborhood size and distance independent descriptors. Distance analyzes on the newly developed descriptors, called DeLL and HoPD, and variations of them show they are independent of the distance between LiDAR sensor and object.

As an additional topic, this chapter discusses existing intensity-based descriptors and introduces a novel intensity-based descriptor, called GRAIL. Because intensity-based descriptors use only the one-dimensional intensity information of LiDAR point clouds compared to the three-dimensional information processed by geometry-based descriptors, this part of the chapter examines the practicability for localization of intensity-based descriptors compared to geometry-based descriptors, the second research question. Odometry calculation results based on both types of descriptors show that they produce relative movements with similar accuracies. However, the standard deviation of the odometry accuracies is larger because the accuracy depends on the occurrence of high-reflectivity information within a scene. Hence, intensity-based descriptors provide sufficient information, which can be used for localization, but not as a stand-alone solution.

The outcome of this first step of the *LiDAR-Feature-based Localization* provides descriptions for every point of the point cloud. However, not every point of the point cloud should be used for localization, as this results in a large amount of data and not every point provides useful information for localization. Therefore, the next step of the *LiDAR-Feature-based Localization* is to use the descriptions to detect those areas with information useful for localization. This is presented in the next chapter.

4. Feature Area Extractions

The first step of the *LiDAR-Feature-based Localization* computes descriptions for every point in the point cloud. However, this leads to large amounts of data and not every point of the point cloud provides robust information which is useful for localization. Hence, in the second step of the *LiDAR-Feature-based Localization*, the point cloud information and the descriptions computed with a fixed, arbitrary descriptor are used to extract areas automatically which fulfill these requirements. This chapter focuses on the extraction with geometry-based descriptors, since intensity-based descriptors need to be handled differently. For the extraction of intensity-based descriptors, see appendix C. It depicts the methodology of the extraction process, defining the criteria for an automated extraction method and the research questions, in Section 4.1. The developments of the extraction process build upon these criteria. The first step of the extraction process is the generation of connected sets of points with similar descriptions, called Feature Areas (FAs), presented in Section 4.2. In the second step of the extraction process, the benefit for localization of each computed FA is measured, refer to Section 4.3. The implementation of these two steps are described in Section 4.4, which also includes the specification of the map data format. The developed extraction process is analyzed and its results are presented in Section 4.5. Especially one experiment investigates an aspects of the practical relevance of the *LiDAR-Feature-based Localization*: the scenery coverage of FAs Subsection 4.5.3.

4.1. Terms and Methodology

The methodology of the detection process is introduced in this section. It includes requirements for developing the extraction of Feature Areas (FAs) and the research questions addressed in this chapter.

4.1.1. Defining the Feature Extraction

In the following subsection, the terms of the second step of the *LiDAR-Feature-based Localization* are defined. This consists in extracting Feature Areas (FAs) with descriptions of 3D LiDAR data which are most useful for localization in an automated way. This is done with on-board sparse point clouds to extract On-board Feature Areas (OFAs) and with globally referenced dense point clouds to extract Map Feature Areas (MFAs). The following definitions are specified for a fixed, arbitrary descriptor function D .

Definition 4.1 (Connected Points). A subset $\mathcal{P}_k^{x,y,z}$ of a 3D point cloud $\mathcal{P}^{x,y,z} \subset \mathbb{R}^3$ is called connected if

$$\text{convex hull}(\mathcal{P}_k^{x,y,z}) \cap \mathcal{P}^{x,y,z} = \mathcal{P}_k^{x,y,z}.$$

Definition 4.2 (Feature Area). Let $\mathcal{P}_k^{x,y,z} \subset \mathcal{P}^{x,y,z} \subset \mathbb{R}^3$ be a connected set of 3D points of the point cloud \mathcal{P} , let $\mathcal{D}_{\mathcal{P}_k} \subset \mathcal{D}_{\mathcal{P}} \subset \mathbb{R}^m$ be the set of descriptions of \mathcal{P}_k with local neighborhood \mathcal{N} , let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some descriptor-specific metric, and let $\varepsilon > 0$ be a predefined metric threshold. Then, a Feature Area (FA) $\mathcal{F}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}^{3D} = (\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k})$ is a set if

$$\forall \mathbf{p}, \mathbf{q} \in \mathcal{P}_k \text{ satisfy } m_d(D_{\mathcal{P}_k}(\mathcal{N}(\mathbf{p})), D(\mathcal{N}(\mathbf{q}))) < \varepsilon.$$

In other words, a Feature Area (FA) is a connected set of points of a point cloud with similar descriptions. The set of all FAs of one point cloud is denoted with $\mathcal{F}_{\mathcal{P}}$ or \mathcal{F} where no risk of confusion exists.

Definition 4.3 (Feature Area Representative). Let $\mathcal{F}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}^{3D} \subset \mathcal{P}_k^{x,y,z} \times \mathcal{D}_{\mathcal{P}_k}$ be a Feature Area with the connected set of 3D points $\mathcal{P}_k^{x,y,z} \subset \mathcal{P}^{x,y,z} \subset \mathbb{R}^3$ of the point cloud \mathcal{P} and descriptions $\mathcal{D}_{\mathcal{P}_k} \subset \mathcal{D}_{\mathcal{P}} \subset \mathbb{R}^m$ of \mathcal{P}_k . Then, a Feature Area Representative (FAR) $\mathcal{F}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}$ is the representation of a $\mathcal{F}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}^{3D}$ and an element

$$(\mathbf{r}, \mathbf{d}) \in \mathbb{R}^{2^+} \times \mathbb{R}^m,$$

where $\mathbf{d} \in \mathbb{R}^m$ is the medoid of all descriptions $\mathcal{D}_{\mathcal{P}_k}$ and $\mathbf{r} \in \mathbb{R}^{2^+}$ a two-dimensional polygonal chain.

The term \mathcal{F} replaces the term $\mathcal{F}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}$ and the index k is omitted of the point subset $\mathcal{P}_k^{x,y,z}$ in cases where no risk of confusion exists.

The polygonal chain representing the points of a FA are derived from the projection of the points to the $x - y$ -plane. A 2D representation is sufficient since this thesis focuses on 2D localization. As LiDAR sensors scan the surface of objects of the environment, the neglect of the depth information is reasonable. That means that the distinction between one-dimensional objects, i.e., local objects, and approximately two-dimensional objects, i.e., elongated objects, is adequate. A polygonal chain is a union of connecting lines of a series of points.

In this thesis, FAs with local expansions in x - and y -direction smaller than 0.5 meters are modeled as a polygonal chains containing only one point, the two-dimensional average of all points of a FA. Otherwise, the areas are considered as polygonal chains with at least two points. The elongated objects are represented as polygonal chains not lines, because these objects are often not formed like straight lines, e.g., curb courses.

However, in some cases, FAs with local expansions in x - and y -direction larger than 0.5 meters may be modeled as points, called center points. These center points are defined the two-dimensional average of all points of a FA. Here, the Feature Area Representative (FAR) is denoted with $\overset{\circ}{\mathcal{F}}_{\mathcal{P}_k, \mathcal{D}_{\mathcal{P}_k}}$.

Next, the function measuring the benefit for localization of each computed FAR is defined.

Definition 4.4 (Localization Information Gain I). Let $\mathcal{F} \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be a Feature Area Representative. The Localization Information Gain (LIG) is a function

$$LIG : \mathcal{F} \in \mathbb{R}^{2^+} \times \mathbb{R}^m \mapsto [0, 1]^k$$

stating the benefit for localization of each computed Feature Area Representative with k measures.

The definition of the Localization Information Gain (LIG) allows the definition of an On-board Feature Area (OFA), i.e., online detections in the test vehicle.

Definition 4.5 (On-Board Feature Areas). Let $\mathcal{F}_{\mathcal{P}_{on-board}, \mathcal{D}_{\mathcal{P}_{on-board}}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ be the set of all Feature Area Representatives of a sparse on-board point cloud $\mathcal{P}_{on-board}$ with descriptions $\mathcal{D}_{\mathcal{P}_{on-board}}$, let LIG be the function measuring the benefit for localization of each computed Feature Area Representative, and let $q_{1-(\alpha/100)}$ be the k -dimensional $(100 - \alpha)\%$ -quantile of the LIG function's distribution of all Feature Area Representatives of the set $\mathcal{P}_{on-board} \times \mathcal{D}_{\mathcal{P}_{on-board}}$ with $\alpha \in [0, 100]$. Then, the set of On-board Feature Areas (OFAs) $\mathcal{F}_{\mathcal{O}}$ is defined by

$$\mathcal{F}_{\mathcal{O}} = \{\mathcal{F}_{\mathcal{P}_{on-board}, \mathcal{D}_{\mathcal{P}_{on-board}}} \in \mathcal{F}_{\mathcal{P}_{on-board}, \mathcal{D}_{\mathcal{P}_{on-board}}} | LIG(\mathcal{F}_{\mathcal{P}_{on-board}, \mathcal{D}_{\mathcal{P}_{on-board}}}) \geq q_{1-(\alpha/100)}\}.$$

Rephrased, those $\alpha\%$ of FARs of an on-board point cloud with the highest LIG score are extracted as OFAs.

Additionally, the definition of the LIG enables the definition of a Map Feature Area (MFA), i.e., offline detections for the map generation.

Definition 4.6 (Map Feature Areas). Let $\mathcal{F}_{\mathcal{P}_{map}, \mathcal{D}_{\mathcal{P}_{map}}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ be the set of all Feature Area Representatives of a dense map point cloud \mathcal{P}_{map} with descriptions $\mathcal{D}_{\mathcal{P}_{map}}$, let LIG be the function measuring the benefit for localization of each computed Feature Area Representative, and let $q_{1-(\alpha/100)}$ be the m -dimensional $(100 - \alpha)\%$ -quantile of the LIG function's distribution of all Feature Area Representatives of the set $\mathcal{P}_{map} \times \mathcal{D}_{\mathcal{P}_{map}}$ with $\alpha \in [0, 100]$. Then, the set of Map Feature Areas (MFAs) $\mathcal{F}_{\mathcal{M}}$ is defined by

$$\mathcal{F}_{\mathcal{M}} = \{\mathcal{F}_{\mathcal{P}_{map}, \mathcal{D}_{\mathcal{P}_{map}}} \in \mathcal{F}_{\mathcal{P}_{map}, \mathcal{D}_{\mathcal{P}_{map}}} | LIG(\mathcal{F}_{\mathcal{P}_{map}, \mathcal{D}_{\mathcal{P}_{map}}}) \geq q_{1-(\alpha/100)}\}.$$

To express it differently, those $\alpha\%$ of FARs of a map point cloud are extracted as MFAs with the highest LIG score.

Summarizing the definitions, the FA is defined as a connected set of points of a map or on-board point cloud with similar descriptions. A FA are represented by a position and a description as FAR. The evaluation function LIG for FARs is used to select the subset of OFAs $\mathcal{F}_{\mathcal{O}}$ and MFAs $\mathcal{F}_{\mathcal{M}}$ by keeping the best α percent and representing them two-dimensional. Depending on the data type, i.e., on-board sampled point clouds or globally referenced point clouds, different aspects of the LIG may be considered.

Thus, the goal of the second step of the *LiDAR-Feature-based Localization* can be formulated as follows.

Problem Formulation:

Let $\mathcal{P} = (\mathcal{P}^{x,y,z}, \mathcal{P}^i) \subset \mathbb{R}^3 \times [0, 1]$ be a measured point cloud with 3D and intensity information and $\mathcal{D}_{\mathcal{P}} \subset \mathbb{R}^m$ be the set of descriptions of that point cloud. The goal is to extract Map Feature Areas $\mathcal{F}_{\mathcal{M}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ of globally referenced point clouds, which can be recognized in the vehicle as On-board Feature Areas $\mathcal{F}_{\mathcal{O}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ of on-board point clouds.

4.1.2. Procedure of Feature Area Extraction

The procedure of the FA extraction approach consists of two steps: the development of a method extracting OFAs and MFAs, and the evaluation of the developed method in the context of *LiDAR-Feature-based Localization*. The methodologies of these steps are explained in detail in the following.

Developing the Extraction Process: A method for extracting OFAs and MFAs shall be developed for their application in the *LiDAR-Feature-based Localization*. Therefore, several requirements in the context of *LiDAR-Feature-based Localization* are specified which the FA extraction algorithm must fulfill.

Requirements 4.7 (Feature Area Extraction Method). A method, extracting On-board Feature Areas $\mathcal{F}_{\mathcal{O}}$ and Map Feature Areas $\mathcal{F}_{\mathcal{M}}$, has to fulfill the following conditions:

The extraction method has to ...

- (i) detect areas \mathcal{F} rather than single key points $p^{x,y,z}$ in 3D LiDAR point clouds \mathcal{P} .
- (ii) detect areas \mathcal{F} based on their descriptions $\mathcal{D}_{\mathcal{P}}$ in 3D LiDAR point clouds \mathcal{P} . This implies that the extraction is not based on semantic interpretation.
- (iii) process real-world data, i.e., real-world 3D LiDAR point clouds \mathcal{P} .
- (iv) detect a variable number of areas \mathcal{F} depending on the LiDAR point cloud \mathcal{P} .

These requirements for the FA extraction algorithms are explained in more detail in the following.

The first requirement states that areas should be detected in contrast to extracting single key points from LiDAR point clouds. In the literature, often key points are extracted based on their large curvature or surface variation values within their local neighborhood, refer to Subsection 2.1.2 [46, 111, 137, 143]. In the context of *LiDAR-Feature-based Localization*, this is a great disadvantage as points on planar surfaces and less curved parts will never be detected. Planar surfaces provide useful localization information along the surfaces' orientations. Additionally, it is difficult to handle a repeatable matching of single key points with map key points using sparse point clouds of LiDAR sensors. Thus, it is favorable to detect FAs, i.e., areas of connected points with similar descriptions, since they can be detected more reliably than single key points. At the same time, by grouping the points with similar descriptions, the data is compressed while maintaining the same information content as key-point-based concepts.

In the second requirement, it is specified that the FAs should be detected in a non-semantic way. Based on the 3D point information and their descriptions, robust and persistent FAs should be extracted.

The extraction method has to cope with description data computed from real-world data. The third requirement captures this. Description data is noisy and incomplete. Thus, it requires suitable algorithms.

The fourth requirement specifies that a variable number of FAs must be detectable with the extraction method. Since the number of robust non-semantic patterns occurring in the environment or a point cloud depends on the environment, methods are favorable whose calculation rule does not require an a priori definition of the number of FAs.

Based on these criteria, clustering algorithms are applied for the extraction concept developed in this chapter similar to the point cloud segmentation methods, refer to Subsection 2.1.2 and Figure 2.7. These enable the descriptions and points of a point cloud to be combined into groups of similar, non-semantic patterns, meeting Requirement item (ii). Additionally, the clustering algorithms provide compression of a large amount of map data, i. e. point cloud and description data, without losing information, fulfilling Requirement (iv). The feature selection concept presented here is a two-stage clustering process. Two clustering phases are combined, which take two different characteristics of the input data into account, i.e., descriptions and point cloud coordinates, to group spatially connected points with similar descriptions, cf. Definition 4.2. Therefore, one clustering step must be carried out in the description space, the other one in the Euclidean space. This means that a first clustering algorithm sorts all points of a point cloud into similar groups based on their multi-dimensional descriptions. A second clustering algorithm orders the points in spatially similar groups with respect to their Euclidean 3D coordinates, i.e., areas and thus fulfilling Requirement item (i). Suitable clustering algorithms that fulfill the other conditions are chosen later in this section, building the basis for the extraction process.

Evaluation of the Developed Extraction Process: The developed extraction method should be evaluated for their application in the *LiDAR-Feature-based Localization*. The evaluation is divided into two parts. First, the developed detection of FAs, i.e., connected points with similar descriptions, is evaluated. Second, the developed detection of OFAs and MFAs, i.e., the on-board and map elements, which are useful for localization, is evaluated.

The clustering algorithms build the basis for the evaluation of the FAs detection, see previous procedure step. They are applied for two reasons: to group similar descriptions and to group close points. Therefore, two different sets of criteria for the evaluation of suitable clustering algorithms are specified in the following.

First, the criteria for the clustering algorithms applied for the description clustering are introduced.

Requirements 4.8 (Clustering in Description Space). *A clustering algorithm, grouping descriptions \mathcal{D} , has to fulfill the following conditions:*

- (i) *Homogeneity: The descriptions within a cluster must be as similar to one another and as dissimilar to descriptions from other clusters as possible.*
- (ii) *Handling High-Dimensional and Noisy Data: A clustering algorithm must be able to process high-dimensional and noisy descriptions $\mathcal{D}_{\mathcal{P}}$.*
- (iii) *Level of Classification: The clusters found must represent the description data $\mathcal{D}_{\mathcal{P}}$ in such a way that no patterns in the environment are lost (underclassification) and the division of a homogeneous set of descriptions into several clusters is avoided (overclassification).*
- (iv) *Uniqueness: All descriptions $\mathcal{D}_{\mathcal{P}}$ are assigned to exactly one cluster.*
- (v) *Outlier Robustness: An elimination of data outliers should be possible.*

These criteria for the clustering algorithms applied in the description space are explained in more detail.

The first requirement is the most important one. A clustering algorithm should enable a clear differentiation between the different description patterns.

Also, the clustering algorithms have to group high-dimensional descriptions computed from real-world data reliably. This second criterion is derived from Requirement 4.7 (iii). For instance, the FPFH, the SHOT, and the RoPS descriptions are 33-, 352-, and 135-dimensional vectors, respectively, refer to Subsection 2.1.1.

Additionally, no relevant information must be lost during the clustering process since clustering algorithms reduce information. That means that inhomogeneous descriptions should not be assigned to the same cluster, i.e., no underclassification, and uniform descriptions should occur in only one common cluster, i.e., no overclassification.

The fourth requirement includes the uniqueness of the cluster allocation, which is essential for extracting clear FAs. Accordingly, hard clustering methods (clear cluster allocation) are preferable to soft clustering methods (cluster allocation based on probability values).

The algorithms must determine cluster assignments regardless of the quality of the description data, captured by the fifth criterion. The presence of data outliers makes it challenging to assign an object to a suitable cluster, which is why the results calculated by an algorithm can be influenced [30]. Therefore, a natural selection of data outliers should be made possible, e.g., by grouping the outliers into a separate cluster.

Second, the criteria for the clustering algorithms applied for the 3D point clustering are introduced.

Requirements 4.9 (Clustering in Euclidean Space). *A clustering algorithm, grouping points of a point cloud \mathcal{P} , has to fulfill the following conditions:*

- (i) *Uniqueness: All points of the 3D LiDAR point cloud \mathcal{P} are assigned to at most one cluster.*
- (ii) *Variable Number of Clusters: The cluster algorithms' calculation rule should enable a variable number of clusters.*
- (iii) *Outlier Robustness: An elimination of data outliers should be possible.*

The previous criteria for the clustering algorithms applied in the Euclidean space are explained in more detail.

The first requirement states that the cluster allocation's should be unique. The concept of *LiDAR-Feature-based Localization* requires the selection of FAs that have a unique position for matching offline generated MFAs with online in the vehicle detected OFAs. For this reason, the points of a point cloud must only be assigned to exactly one cluster. Thus, hard clustering procedures (clear cluster allocation) are preferable to soft clustering procedures (cluster allocation based on probability values).

The second requirement is derived from Requirements 4.7 (iv). The number of FAs is unknown a priori. Thus, the clustering algorithm grouping areas must be able to detect a variable number of Euclidean clusters.

In the third requirement, it is specified that a clustering algorithm should be capable to group real-world 3D LiDAR point clouds reliably independent of their quality. Therefore, a selection of data outliers should be enabled by clustering algorithms.

These criteria are used to choose suitable clustering algorithms for the application in the description and Euclidean space to detect FAs. Selected clustering algorithms are evaluated based on each criterion to choose a combination of two clustering algorithms. These clustering algorithms form the basis for the generation of FAs.

The second evaluation step is assessing the detection of OFAs and MFAs, i.e., the on-board and map elements, which are useful for localization. Therefore, the criteria for evaluating the extraction process of OFAs and MFAs are defined in the following.

Requirements 4.10 (Extraction of On-Board and Map Feature Areas). *An extraction algorithm, detecting OFAs and MFAs, has to fulfill the following conditions:*

- (i) *Usefulness: The extracted OFAs \mathcal{F}_O and MFAs \mathcal{F}_M are useful for localization.*
- (ii) *Local Assignability: The extracted OFAs \mathcal{F}_O should change moderately from frame to frame during a test drive.*
- (iii) *Global Assignability: The extracted OFAs \mathcal{F}_O and MFAs \mathcal{F}_M should match. Ideally, the set of OFAs should be a proper subset of the set of MFAs.*
- (iv) *Compressibility: The extraction process has to compress the input data, i.e., point clouds \mathcal{P} and descriptions \mathcal{D}_P , reducing the memory size of the map, i.e., MFAs \mathcal{F}_M .*

These criteria for the extraction method applied for OFAs and MFAs are explained in more detail.

The first requirement defines that only those areas should be extracted which are beneficial for localization. The main task of feature selection is to differentiate between points of a point cloud that are suitable and that are not relevant for localization. This measure is called Localization Information Gain (LIG), refer to Definition 4.4.

In the second requirement, it is specified that in each test drive, enough similar OFAs are extracted. The OFAs should only change moderately. Ideally, these test drives should be carried out under various conditions, e.g., different weather conditions.

The third requirement states that the extracted OFAs and MFAs should match. This matching is the essential step of the map-based localization and is therefore crucial. Additionally, the set of MFAs should contain all elements of the set of OFAs to enable a robust matching.

The last requirement describes that the extraction method should compress the input data to reduce the map's memory size. Thus, the input point cloud's memory size and the input description data should be reduced by building a compressed map.

These criteria are used to evaluate the process of selecting OFAs and MFAs from the set of FAs.

Data Sets: For the evaluation of selecting OFAs and MFAs from the set of FARs, two data set types of LiDAR point clouds and the resulting descriptions are used as input data for the extraction. Therefore,

real-world point clouds, recorded with close-to-production-type LiDAR sensors and recorded and accumulated in postprocessing with a mobile data acquisition system, are considered. Both types are described in the previous chapter, refer to Section 3.1, and can be seen in Figure 3.2b and in Figure 3.2c.

On the one hand, data sets from close-to-production Velodyne VLP 16 [151] and Velodyne VLP 32-C [150] are used. These sensors scan the surroundings with a non-uniform resolution as the horizontal resolution is higher than the vertical resolution, see Figure 3.2. The point clouds measured with these sensors are noisy and sparse. Besides, the scanning density decreases with increasing distance between the sensor and the scanned environmental objects. Because of this, the data captures only an incomplete representation of reality and is subject to measurement inaccuracies. This point cloud data $\mathcal{P}_{\text{on-board}}$ forms the basis for the extraction of OFAs.

On the other hand, point clouds are used that were created by a mobile data acquisition system [105, 142]. The Trimble MX8 sensor, which is integrated into this system, measures the environment with high precision by repeated scanning and postprocessing to generate a globally referenced dense point cloud with a sampling resolution of approximately $34 \frac{\text{points}}{100 \text{ cm}^2}$. Resulting point clouds build a complete and ideal representation of the reality. This point cloud data \mathcal{P}_{map} forms the basis for the extraction of MFAs, i.e., for the map generation.

These data sets are used to test, evaluate, and to determine parameters of the FAs extraction process.

4.1.3. Research Questions of the Application of Feature Extraction

In the following, two research questions are presented and explained which are addressed in this chapter.

Which methods are well-suited to detect good Feature Areas?

- As stated in the previous subsection, the choice of well-suited methods to detect good FAs depends on different criteria. For that, clustering algorithms are depicted to extract OFAs and MFAs in a non-semantic way. The main difficulty in this thesis when clustering descriptions lies in their high-dimensionality and noisiness. During clustering, often cluster representatives of descriptions are computed to which the elements of the input data are associated. This is complex for high-dimensional and noisy data.
- Several clustering algorithms are evaluated based on the criteria for description and point cloud clustering to provide an answer to this research question, refer to Requirements 4.8 and Requirements 4.9. These evaluations serve to select the clustering algorithms used in this thesis for the detection of FAs.

Which properties are well-suited to select good Feature Areas?

- OFAs and MFAs have to be selected from the set of FAs to enable the detection of robust areas for a reliable localization. Suitable characteristics must be developed to evaluate the benefit of these areas for localization. This measure is called Localization Information Gain (LIG), refer to Definition 4.4.
- A distinction between OFAs and MFAs must be made when evaluating their benefit for localization. As the OFAs should be ideally a proper subset of MFAs, i.e., MFAs should contain all elements of OFAs, different aspects of the LIG must be considered, depending on the data type, i.e., on-board sampled point clouds or point clouds used for the map generation.

One paper presenting investigations on the preceding research questions has been published which resulted from this work. In [53], the developed extraction method is presented and evaluated. This paper also includes the definition of the measure LIG measuring the benefit for localization.

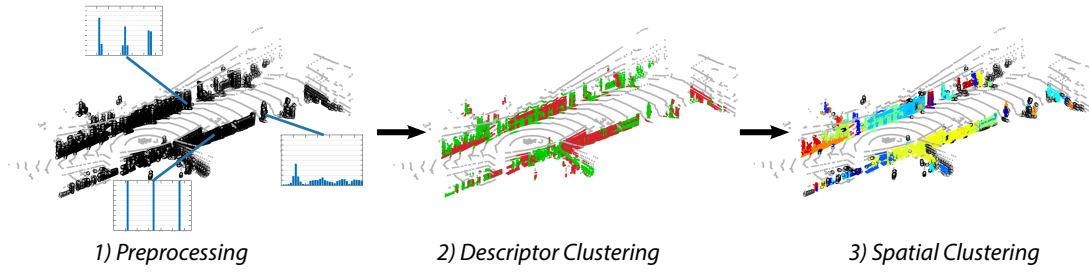


Figure 4.1: Scheme of the proposed FA generation method. 1) Suitable points of the point cloud (colored in black) are selected and used for the description calculation (white-blue bar charts). 2) By clustering the point’s descriptions, differentiation between two classes (green, red) is made, and 3) cluster those groups spatially (points belonging to the same cluster are marked with one color). Based on [53].

4.2. Generation of Feature Areas in Non-Semantic Way

In the following, the method of the detection of FAs, i.e., connected points of a LiDAR point cloud with similar geometry-based descriptions, is presented. The main idea of the FAs generation is a two-step clustering approach considering both the point cloud and their descriptions. These unsupervised algorithms are applied as non-semantic elements should be extracted in the environment, refer to Subsection 4.1.2.

The FA generation consists of three steps. First, only descriptions of points in the input point cloud with enough points in the local neighborhood are calculated. The next two steps are the two-stage clustering approach taking the descriptions and the point’s euclidean coordinates as input data into account. Therefore, points with similar descriptions are grouped as a first clustering step. Afterward, description clusters are grouped into areas containing spatially connected points. This results in multiple spatially connected areas, i.e., non-semantic objects, with similar descriptions - FAs. Figure 4.1 illustrates the three steps. In the following, those three steps are depicted in more detail in separate subsections.

4.2.1. Preprocessing

At first, those points of the input point cloud are selected which provide suitable information for the description calculation and localization.

Those points are chosen whose local neighborhood includes enough points to characterize the three-dimensional structure with the descriptions. For on-board point clouds, this is ensured if the local neighborhood is large enough, refer to Definition 3.5. The horizontal resolution is often higher than the vertical resolution of close-to-production type LiDAR sensors. Therefore, if the neighborhood encloses enough points in the vertical direction to guarantee a three-dimensional structure, it encloses enough points in the horizontal resolution. It follows from this that a local neighborhood with a large enough radius must contain points at least of one layer above and below the query point [150, 151]. Additionally, points are selected for the description calculation if the number of points falling into the local neighborhood $\mathcal{N}_{\mathcal{P}_{\text{on-board}}, r}(\mathbf{p}^{x,y,z})$ around point \mathbf{p} exceeds a threshold λ . This yields the subset of preprocessed on-board points $\mathcal{P}_{\text{preproc on-board}} \subset \mathcal{P}_{\text{on-board}}$, for which a description is computed.

$$\mathcal{P}_{\text{preproc on-board}} = \{\mathbf{p} \in \mathcal{P}_{\text{on-board}} \mid |\mathcal{N}_{\mathcal{P}_{\text{on-board}}, r}(\mathbf{p}^{x,y,z})| > \lambda \wedge r \text{ is large enough}\} \quad (4.1)$$

Practical experiments on real-world data have shown that, for sparse on-board point clouds, a λ of 12 and, for dense map point clouds, a λ of 20 is suitable, refer to Subsection 4.1.2. Especially, the resolution of histogram-based descriptions, e.g., the FPFH and the SHOT, largely depend on the number of points in the local neighborhood. If their local neighborhood or parts of their local neighborhood encloses only a few points, the descriptions' precision of the numerical values is low. In this case, the descriptions are too imprecise to later yield reliable clustering results. Hence, 4.1 results in the subset of points used for the description computation of OFAs.

Regarding dense map point clouds of LiDAR sensors, the sampling is uniform. Thus, a point's p description is determined if the number of points falling into the local neighborhood $\mathcal{N}_{\mathcal{P}_{\text{on-board}}, r}(\mathbf{p}^{x,y,z})$ exceeds the threshold λ . This yields the subset of preprocessed map points $\mathcal{P}_{\text{preproc, map}} \subset \mathcal{P}_{\text{map}}$, for which a description is computed.

$$\mathcal{P}_{\text{preproc, map}} = \{\mathbf{p} \in \mathcal{P}_{\text{map}} \mid |\mathcal{N}_{\mathcal{P}_{\text{map}}, r}(\mathbf{p}^{x,y,z})| > \lambda\} \quad (4.2)$$

Prior to the description clustering, further points of the preprocessed map and on-board point cloud $\mathcal{P}_{\text{preproc, map, on-board}}$ are rejected. As points lying on horizontal planes do not provide information for 2D localization, points with this characteristic are discarded. The subset $\mathcal{P}_{\text{preproc}' \text{ map, on-board}}$ is computed for the preprocessed map and on-board point cloud $\mathcal{P}_{\text{preproc, map, on-board}}$ by

$$\mathcal{P}_{\text{preproc}' \text{ map, on-board}} = \{\mathbf{p} \in \mathcal{P}_{\text{preproc, map, on-board}} \mid \angle(z_{\text{VRF}}, \mathbf{n}_p) \geq 20^\circ\}, \quad (4.3)$$

with $\angle(z_{\text{VRF}}, \mathbf{n}_p)$ being the angle between the height-axis of the VRF and the normal \mathbf{n}_p of the query point p . Choosing a threshold of 20° has proven to be suitable while dealing with real-world data, refer to Subsection 4.1.2.

This results in the subset of points used for the description computation of MFAs.

This selection process is crucial regarding the following clustering algorithms. As clustering algorithms consider the input data as a whole and descriptions are usually multi-dimensional vectors highly sensitive to noise, imprecise data influence the clustering results a lot when computing cluster representatives. These cluster representatives form often the basis of the clustering algorithm. Commonly, during clustering, the elements of the input data are associated to the clustering representatives. Imprecise data change these cluster representatives to be more similar to the imprecise data. Therefore, the association to cluster representatives, which are similar to imprecise data, affects the clustering results.

For the resulting point cloud subsets $\mathcal{P}_{\text{preproc}' \text{ map, on-board}}$, the descriptions are calculated with the descriptors selected in Chapter 3, respectively, i.e., FPFH, SHOT, RoPS. However, the whole point clouds $\mathcal{P}_{\text{map, on-board}}$ are used for the description calculation of the query points of $\mathcal{P}_{\text{preproc}' \text{ map, on-board}}$. This is the case since some points are not in $\mathcal{P}_{\text{preproc}' \text{ map, on-board}}$ but in the query points local neighborhood.

4.2.2. Description Clustering

In this second step, a clustering step in the description space is carried out. The descriptions are calculated with one given descriptor. That means that the description clustering is performed for each descriptor independently but can be performed for multiple descriptors in parallel. This clustering step uses descriptions as input data. It is done before the spatial clustering. To identify different non-semantic objects it is vital to know points characterized by similar patterns. Starting with spatial clustering, the spatial affiliation of the points would be prioritized above the similarity of their associated descriptions, although objects should be detected based on their characteristics, not their spatial expansion. Therefore, the clustering in the description space is performed first.

This means that a clustering algorithm groups all points of a point cloud into similar groups based on their multi-dimensional descriptions. Points with similar descriptions are clustered to compress the

data by maintaining most of the information content. Thus, the amount of memory and the search space for matching OFAs with MFAs is reduced, refer to Requirement (iii). The descriptions are used to differentiate between various non-semantic patterns within the subsets $\mathcal{P}_{\text{preproc map, on-board}}$ based on geometric information.

As a result, there is a set of clusters with descriptions of the same pattern in the environment scanned by the LiDAR sensor. For instance, a cluster consists of all points whose FPFH descriptions characterize a planar geometry depending on the local neighborhood.

Hereafter, further discarding steps can be made to reduce the clusters to the most reliable ones.

As one criterion, the strongly structured clusters from all generated description clusters are kept for further processing. Strongly structured clusters are clusters whose descriptions are similar to one another and dissimilar to that of other clusters. This is explained in more detail in the following.

The description characteristics are identified by calculating the silhouette value s , which is defined as follows [59].

Definition 4.11 (Description's Silhouette Value). Let \mathbf{d}_i be the i -th description of a cluster, let $\delta_{\mathbf{d}_i, \text{intra}}$ be the average distance from the i -th description of a cluster to other elements of the same cluster, and let $\delta_{\mathbf{d}_i, \text{inter}}$ be the minimum of the average distances from the i -th description of one cluster to all elements of any other cluster. Then, the silhouette value of the description $s(\mathbf{d}_i)$ is defined as

$$s(\mathbf{d}_i) = \frac{\delta_{\mathbf{d}_i, \text{inter}} - \delta_{\mathbf{d}_i, \text{intra}}}{\max(\delta_{\mathbf{d}_i, \text{intra}}, \delta_{\mathbf{d}_i, \text{inter}})}. \quad (4.4)$$

The silhouette value ranges from -1 to +1. A silhouette value close to one indicates that the element is well-matched to its cluster and poorly-matched to neighboring clusters. Values close to zero indicate that the intra- and intercluster distance of the element are the same. Then, the element is equidistant from two clusters and its assignment is indifferent. If the silhouette coefficient becomes negative, the intracluster distance is greater than the intercluster distance. The element is then, on average, more similar to another cluster than its own.

The definition of the description's silhouette value enables the definition of the cluster's silhouette value. [59]

Definition 4.12 (Cluster's Silhouette Value). Let \mathcal{C} be a cluster containing l descriptions $\mathcal{D} \subset \mathbb{R}^m$ with silhouette values $s(\mathbf{d}_i)$, where $\mathbf{d}_i \in \mathcal{D}$ denotes the i -th description. Then, the silhouette value of cluster \mathcal{C} is defined as the average of its descriptions' silhouette value:

$$\bar{s}(\mathcal{C}) = \frac{1}{l} \cdot \sum_{i=1}^l s(\mathbf{d}_i). \quad (4.5)$$

This enables the definition of strongly structured clusters.

Definition 4.13 (Strongly Structured Cluster). Let \mathcal{C} be a cluster of descriptions and $\bar{s}(\mathcal{C})$ its silhouette value. Then, a cluster is called strongly structured, if it fulfills the following condition:

$$\bar{s}(\mathcal{C}) \geq 0. \quad (4.6)$$

A silhouette value which is greater or equal zero indicates that the cluster's descriptions are well-matched to its cluster and poorly-matched to neighboring clusters. This threshold is set to zero since in practical experiments zero has been observed to be appropriate in noisy real-world data as it represents an adequate trade-off between keeping noisy clusters but rejecting clusters which can not be separated accurately, refer to Subsection 4.1.2.

The subset of all strongly structured clusters from all description clusters is kept for further processing. As the descriptions of one description cluster are similar, only a representative is kept to compress the high-dimensional description data. Therefore, the medoid of all descriptions of a cluster is used as part of the FAR, describing the characteristics of the cluster the best, cf. Definition 4.3.

The outcome of the description clustering into two clusters can be seen in Figure 4.1 on the example of the FPFH descriptor.

4.2.3. Spatial Clustering

Subsequently, spatial clustering is carried out for each strongly structured cluster separately. For this purpose, the Euclidean coordinates of the preprocessed point cloud $\mathcal{P}_{\text{preproc' map, on-board}}$ are used as the input for the spatial clustering. The points of the strongly structured description clusters are grouped separately into multiple areas of spatially connected points. For example, from a cluster that contains points with similar FPFH descriptions, several clusters can be formed that spatially divide these points of similar patterns.

This step is crucial as localization requires this spatial information, which is used for two reasons. On the one hand, the spatial information is required to realize a local association, i.e., enabling a repeatable allocation of on-board detected clusters over multiple process steps of localization. On the other hand, the spatial information is required to realize a global association matching on-board elements with map elements.

The obtained clusters require the calculation of new FAR in the description space that best describe each spatial cluster through its description. For this purpose, the medoid of all descriptions of a cluster is computed as part of the FAR, cf. Definition 4.3. As spatial representative, a polygonal chain is computed. Details about the FAR implementation are provided in Section 4.3.

In summary, this spatial clustering outputs FAs. The results of this clustering step through the example for an on-board point cloud can be seen in Figure 4.1.

4.3. Extraction of Map and On-Board FAs applying the LIG

In this section, the definition of the components of the Localization Information Gain (LIG) are presented. The LIG measures the benefit for localization of each FAR, representing the FAs, refer to Definition 4.4. This section answers the second research question of this chapter: Which properties are well-suited to detect good feature areas, cf. Subsection 4.1.3?

The main idea of the LIG determination is the definition of five aspects which capture different evaluation criteria of localization. These are used to keep the best $\alpha\%$ of FAs, thus extracting OFAs and MFAs. The LIG is designed to fulfill Requirements 4.10 (i), (ii), and (iii), i.e., keeping only those FAs which are useful for localization, which are recognizable through several test drives, and whose on-board detections match those in the map.

The following five criteria are defined to compose the LIG. The first three are defined for FARs extracted based on the data of one LiDAR scan and the last two are defined for FARs extracted based on the data of multiple LiDAR scans. Therefore, the first three are defined formally, the last two are defined informally as their parameter space is too large for a comprehensible definition.

Definition 4.14 (Distinctiveness). For each descriptor D , $LIG_{d,D}$ is a function

$$LIG_{d,D} : \mathbb{R}^{2^+} \times \mathbb{R}^m \rightarrow [0, 1].$$

It assigns a value to a Feature Area Representative which captures how rare and distinguishable the description of a Feature Area Representative is.

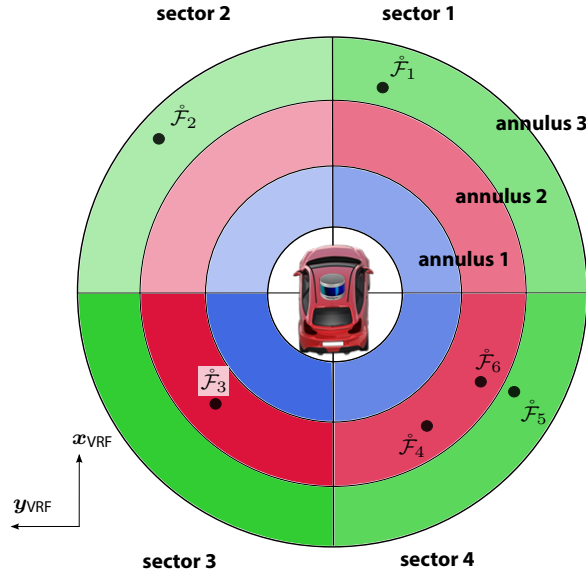


Figure 4.2: The division of the environment scanned by the LiDAR sensor into four circular sectors (different transparency) and three concentric annuli (blue, red, green). The number of FARs $\hat{\mathcal{F}}_i$ in each circular sector and annulus yields the spatial diversity.

The distinctiveness $LIG_{d,D}$ is defined with a descriptor-specific measure. E. g. the measure captures the non-uniformity of the FAR's component values of its description. That means the distinctiveness is low for a FAR with a description of similar values and higher for vectors with peaks. For instance, a low distinctiveness is the case for scattered objects, like plants.

Definition 4.15 (Uniqueness). Let $d_i \in \mathcal{D}_{\mathcal{P}_{on-board}}$ be a description of the i -th Feature Area Representative \mathcal{F}_i of an on-board point cloud $P_{on-board}$ of n points computed with a given descriptor D , let d_{max} be the maximum distance between two descriptions, and let $\mathbf{D}_i = [m_d(d_1, d_i) \dots m_d(d_n, d_i)]$ be the i -th row of the description distance matrix of all descriptions within the on-board point cloud with a descriptor-specific metric $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ and $\bar{\mathbf{D}}_i$ its average. Then, for each descriptor D , the uniqueness $LIG_{u,D}$ of Feature Area Representative \mathcal{F}_i is a function

$$LIG_{u,D} : \mathbb{R}^{2+} \times \mathbb{R}^m \rightarrow [0, 1], \mathcal{F}_i \mapsto \frac{\bar{\mathbf{D}}_i}{d_{max}},$$

stating how unique its description is within the on-board point cloud.

The uniqueness is defined as the normalized mean distance of the FAR's description to each FAR's description within a scene, i.e., an on-board LiDAR scan. A uniqueness of a FAR close to zero indicates that similar descriptions to the FAR's description are within the scene, and a uniqueness of a FAR close to one indicates that the FAR's description differs a lot from the other FARs' descriptions in the scene. A FAR can be easily locally and globally assigned if its description differs a lot from the other descriptions within the on-board point cloud, refer to Requirements 4.10 (ii) and (iii).

Definition 4.16 (Spatial Diversity). Let $d_i \in \mathcal{D}_{\mathcal{P}_{on-board}}$ be the description of the i -th Feature Area Representative \mathcal{F}_i of an on-board point cloud $P_{on-board}$ containing l FAs, let the on-board point cloud be divided into j uniform and concentric annuli and k equally sized circular sectors, let $n_{i,ann}$ be the number of other

FARs in the annuli of the i -th FAR \mathring{F}_i , and let $n_{i,cir}$ be the number of other FARs in the circular sector of the i -th FAR \mathring{F}_i . Then, for each descriptor D , the spatial diversity $LIG_{s,D}$ of Feature Area Representative \mathring{F}_i is a measure

$$LIG_{s,D} : \{1, \dots, l\} \times (\mathbb{R}^2 \times \mathbb{R}^m)^l \rightarrow [0, 1], (i, \mathring{F}_i) \mapsto \frac{1}{2} \cdot \left(\frac{1}{n_{i,ann}} + \frac{1}{n_{i,cir}} \right) \forall 1 \leq i \leq l,$$

stating the spatial distribution of FARs within the on-board point cloud.

For an accurate localization, the detections used for the localization task need to be spatially distributed within a scene, i.e., an on-board LiDAR scan. This is represented with the value called spatial diversity $LIG_{s,D}$. In the case of 2D localization, the FARs have to be geographically dispersed in 2D. Depending on the number of FARs within an annulus n_{ann} or circular sector n_{cir} a FAR benefits the spatial diversity. The spatial diversity is high if a FAR is the only one in an annulus and circular sector and low if a FAR is one of multiple FARs within one of them. Figure 4.2 visualizes the division of a point cloud into four circular sectors and three concentric annuli. Depending on the maximum distant point of a point cloud, the circle is divided into a close, middle, and long-range annulus. For example, the spatial diversity of \mathring{F}_5 , in sector 4 with two other FARs on the outer annuli, equals $\frac{1}{2} \cdot \left(\frac{1}{3} + \frac{1}{3} \right) = 0.33$ and the spatial diversity of \mathring{F}_2 , the only FAR in sector 2 on the outer annulus, equals $\frac{1}{2} \cdot \left(\frac{1}{1} + \frac{1}{3} \right) = 0.67$.

The following two LIG components are defined informally, as described above.

Definition 4.17 (Tracking Stability). For each descriptor D , $LIG_{t,D}$ is a function

$$LIG_{t,D} : (\mathbb{R}^{2+} \times \mathbb{R}^m)^n \rightarrow [0, 1].$$

It assigns a value to a Feature Area Representative which captures the average immutability and durability over a short time or small vehicle movements.

The tracking stability $LIG_{t,D}$ specifies the FAR's robust descriptiveness of different points of view over a short time or small vehicle movements, like frame-to-frame changes. Here, it is computed as the mean changes of the FAR's description and computed position. It is also possible to weight the description and the position component of the tracking stability. A low tracking stability suggests that the FAR changes a lot, e.g., a dynamic object changing its position. A high tracking stability suggests that the FAR does not change its position or description.

Definition 4.18 (Persistence). For each descriptor D , $LIG_{p,D}$ is a function

$$LIG_{p,D} : (\mathbb{R}^{2+} \times \mathbb{R}^m)^n \rightarrow [0, 1].$$

It assigns a value to a Feature Area Representative which captures the average immutability and durability over over a long time.

The persistence $LIG_{p,D}$ is defined as the FAR's immutability and durability over a long time, e.g., one year. It measures the relative frequency of OFAs matching the same MFA extracted with data of test drives carried out over a long time. If the persistence is close to one, the same OFA was extracted over a long time. If it is close to zero, the FAR was rarely extracted.

The previous five definitions enable the practical definition of the Localization Information Gain (LIG). Note that the following three LIG definitions are specified informally as the parameter space is very large, which would lead to incomprehensible definitions.

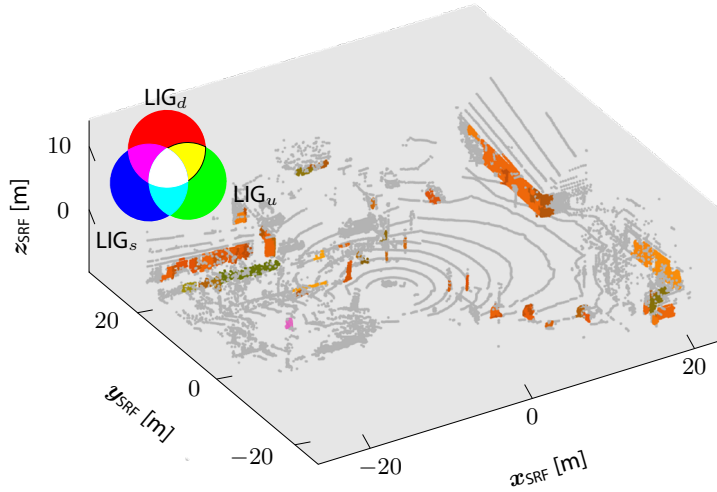


Figure 4.3: Suitable points of an on-board point cloud scanned with a Velodyne VLP 32-C are colored with respect to on-board LIG values using the additive RGB color space [53]. This example is based on the PPFH descriptor [110].

Definition 4.19 (Localization Information Gain II). Let $\mathcal{F}_i \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be the i -th Feature Area Representative of a point cloud, let $LIG_{d,D}(\mathcal{F}_i)$ be the distinctiveness of \mathcal{F}_i , let $LIG_{u,D}(\mathcal{F}_i)$ be the uniqueness of \mathcal{F}_i , let $LIG_{s,D}(\mathcal{F}_i)$ be the spatial diversity of \mathcal{F}_i , let $LIG_{t,D}(\mathcal{F}_i)$ be the tracking stability of \mathcal{F}_i , and let $LIG_{p,D}(\mathcal{F}_i)$ be the persistence of \mathcal{F}_i . Then, the function Localization Information Gain (LIG) of Feature Area Representative \mathcal{F}_i consists of five components

$$LIG : \mathbb{R}^{2^+} \times \mathbb{R}^m \rightarrow [0, 1]^5$$

$$\mathcal{F}_i \mapsto [LIG_{d,D}(\mathcal{F}_i), LIG_{u,D}(\mathcal{F}_i), LIG_{s,D}(\mathcal{F}_i), LIG_{t,D}(\mathcal{F}_i), LIG_{p,D}(\mathcal{F}_i)]^\top$$

stating the benefit for localization of each computed Feature Area Representative.

The LIG components tracking stability and persistence require data collected over a small or a long time using multiple point clouds, respectively. The tracking stability is applied when performing localization and not in the extraction process. This is done in the actual localization so as to not only consider tracking stable OFAs, but rather tracking stable matches of OFAs and MFA. The determination of the persistence is not computable on-board. Therefore, in this thesis, it is suggested to use the persistence only for evaluations of on-board data.

Depending on the data type, i.e., on-board sampled point clouds or map data, different aspects of the remaining LIG components are considered to extract the best $\alpha\%$ depending on their LIG value as OFAs and MFAs, refer to Definition 4.5 and Definition 4.6. Therefore, for both data types, the specific LIG is defined in the following.

Definition 4.20 (On-Board Localization Information Gain). Let $\mathcal{F}_i \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be the i -th Feature Area Representative of an on-board point cloud, let $LIG_{d,D}(\mathcal{F}_i)$ be the distinctiveness of \mathcal{F}_i , let $LIG_{u,D}(\mathcal{F}_i)$ be the uniqueness of \mathcal{F}_i , and let $LIG_{s,D}(\mathcal{F}_i)$ be the spatial diversity of \mathcal{F}_i . Then, the on-board Localization Information Gain of Feature Area Representative \mathcal{F}_i consists of three components

$$LIG_{on-board} : \mathbb{R}^{2^+} \times \mathbb{R}^m \rightarrow [0, 1]^3$$

$$\mathcal{F}_i \mapsto [LIG_{d,D}(\mathcal{F}_i), LIG_{u,D}(\mathcal{F}_i), LIG_{s,D}(\mathcal{F}_i)]^\top$$

stating the benefit for localization of each on-board computed Feature Area Representative.

For extracting OFAs, the consideration of the distinctiveness, the uniqueness, and the spatial diversity is crucial. For instance, in the case of blockage in the on-board scan, FAs with high LIG scores could be blocked, leading to the extraction of FAs with comparably lower scores.

See Figure 4.3 for a colored representation of the LIG values of selected OFAs. The point cloud was sampled with a Velodyne VLP 32-C, and descriptions were computed with the FPFH descriptor. Here, 60% of the FAs with the highest LIG score are extracted as OFAs, equally considering every LIG component, i.e., $\sum_{j \in \{d,u,s\}} \text{LIG}_{j,D}(\mathcal{F})$ being greater than or equals the 60%-quantile. Each of the three LIG components is represented by one color component in the additive RGB space. The distinctiveness is represented with the red component, the uniqueness with the green component, and the spatial diversity with the blue component. As the LIG components are normalized to $[0, 1]$, the percentage of each color corresponds to the value of the LIG component. Because there are many walls in the scene of Figure 4.3, their uniqueness is low. Thus, the green component of the walls is small. However, there is little vegetation in the scene and, therefore, the green component is large. The pillar (pink) is far away from other OFAs, i.e., it is spatially diverse. Hence, the pillar has a large blue component. Additionally, the pillar is very distinctive and, thus, the red component is large. This results in a pink color of the pillar.

Definition 4.21 (Map Localization Information Gain). Let $\mathcal{F} \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be a Feature Area Representative of a map point cloud and let $\text{LIG}_{d,D}(\mathcal{F})$ be the distinctiveness of \mathcal{F} . Then, the measure Localization Information Gain (LIG) of Feature Area Representative \mathcal{F} consists of one component

$$\begin{aligned} \text{LIG} : \mathbb{R}^{2^+} \times \mathbb{R}^m &\rightarrow [0, 1] \\ \mathcal{F}_i &\mapsto [\text{LIG}_{d,D}(\mathcal{F})] \end{aligned}$$

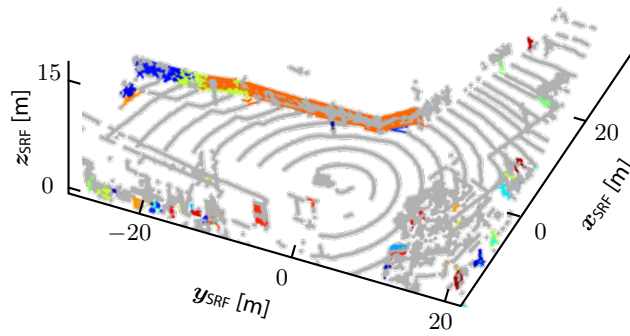
stating the benefit for localization of each computed Feature Area Representative.

When extracting MFAs from globally referenced dense point clouds, only the distinctiveness $\text{LIG}_{d,D}$ is taken into account as only those FAs with good description characteristics should be kept. The goal of the extraction is to fulfill the global assignability of Requirement 4.10 (iii), i.e., to generate MFAs as a proper subset of OFAs. Therefore, only globally valid LIG values are taken into account scene-dependent measures, i.e., neither spatial diversity nor uniqueness. Persistence may be included into the map. However, in this thesis, the map data is considered separately from the on-board data.

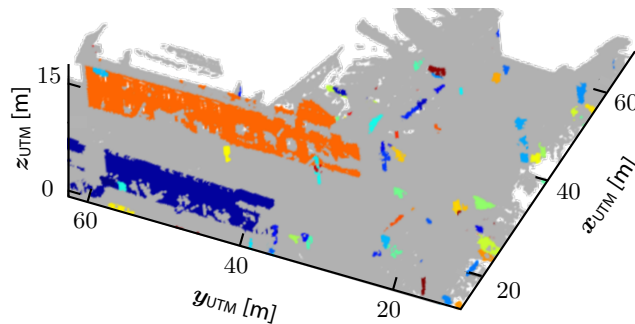
Figure 4.4 shows sample OFAs and MFAs extracted from a sparse and globally referenced dense point cloud in a suburb with descriptors computed with the FPFH descriptor. Here, 60% of the FAs with the highest LIG value are extracted as OFAs and MFAs. Only the distinctiveness for the MFAs is taken into account and the distinctiveness, uniqueness, and spatial diversity are taken into account for the LIG. The points belonging to the OFAs and MFAs are marked in the same color when they belong to the same OFAs or MFAs and the other way around. In this example, mainly walls and pillars are extracted.

After the extraction of the best FAs as OFAs and MFAs with the LIG, the areas' 2D representation is determined. Since the focus of this thesis lies in 2D localization, the selected OFAs and MFAs are represented as either a point or a polygonal chain. MFAs or OFA with local expansions in x - and y -direction smaller than 0.5 meters are modeled as points, i.e., the arithmetic mean of all points of FAs. In practice, 0.5 meters have been proven to provide a sufficient distinction between locally extended areas and elongated objects. Otherwise, areas are considered as polygonal chains with the Ramer-Douglas-Peucker algorithm [27, 103], refer to Subsection 2.2.4. This distinction is reasonable since geometric information of the environment is used for the description calculation. If areas cannot be represented as polygonal chains, often there is a change in their curvature, which is reflected by a change of descriptions. In the first clustering step, the description clustering, refer to 4.2.2, points belonging to those areas are already assigned to different description clusters.

Figure 4.5 shows sample 2D representations of MFAs and OFAs in a suburb. Note that in the figure the



(a) On-board features sampled with a Velodyne VLP 32-C. Only significant features are selected.



(b) Map features generated with a globally referenced, dense point cloud. Only distinctive features are selected.

Figure 4.4: Sample OFAs and MFAs (colored areas) in a sparse and dense point cloud (gray) in a suburb using the FPFH descriptor [110]. Only FAs with distinctive descriptions are selected as MFAs and with distinctive, unique, and spatial diverse characteristics are selected as OFAs. The dense point cloud is plotted in the shifted UTM coordinate system, see Subsection 3.1.2.

corresponding MFAs and OFAs do not have the same color. The descriptions used for this extraction are calculated with the FPFH descriptor. It can be seen that the walls on the left and right side of the street are extracted in the dense map point cloud as well as in the sparse on-board point cloud. For example, while the wall on the right side in the dense map point cloud are extracted as one MFA (orange), they are extracted as multiple OFAs in the on-board point cloud (orange, green, and blue). Due to the on-board point cloud's sparsity, the wall extracted as OFAs is separated into multiple parts. Some artifacts, like parking vehicles, are also extracted, which are not suitable for localization, but hard to distinguish e.g., from walls based on their geometric information.

4.4. Implementation of Extraction Process

In the following section, the algorithms for the extraction process introduced in Section 4.2 are selected based on the criteria defined in Subsection 4.1.2 to answer the first research question of this chapter, cf. Subsection 4.1.3. I. e. it is investigated which methods are well-suited to extract good feature areas. The extraction process consists of two main steps, clustering first the description data and second the description clusters separately in the Euclidean space. Therefore, clustering algorithms for both spaces

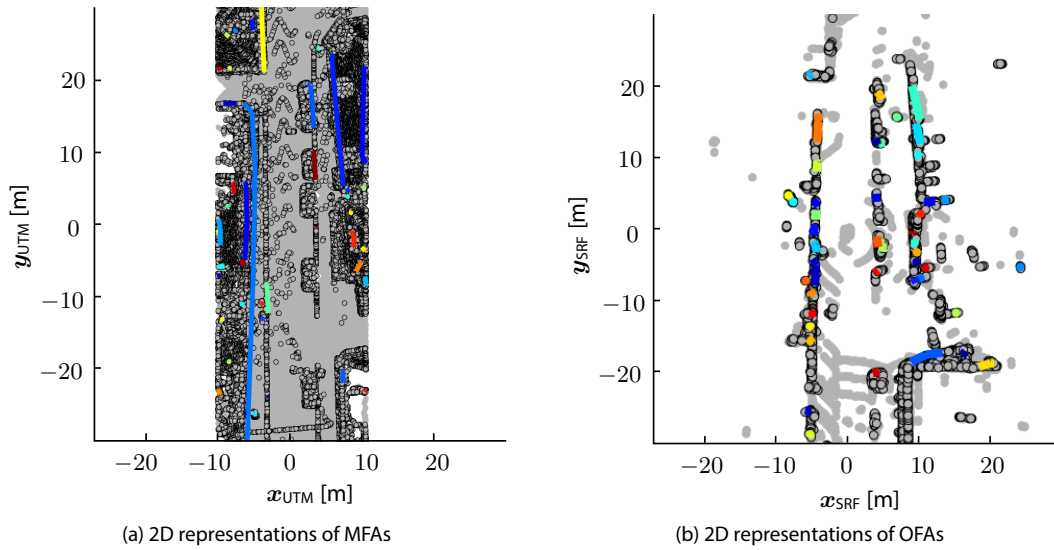


Figure 4.5: Sample 2D representations of MFAs and OFAs (colored points and polygonal chains) in a dense and sparse point cloud (gray) in a suburb. Suitable points (black circles) are used previously as the input for the FPFH description calculation [110]. The dense point cloud is plotted in the shifted UTM coordinate system, see Subsection 3.1.2. The dense and sparse real-world data capture the same section of the environment.

are selected in the following subsections. Additionally, the implementation of the descriptor-specific LIG value called distinctiveness, cf. Definition 4.14, is introduced as well as the map format developed in this thesis.

4.4.1. Implementation of Description Clustering

The first step of the extraction process after the preprocessing is the clustering of the description data. In the following, clustering algorithms for description clustering are compared based on the list of Requirements 4.8. There exist five groups of clustering algorithms: partitioning, hierarchical, density-based, network-based, and model-based [30, 152]. According to Fahad et al. [30] and Venkatkumar and Shardaben [152], hierarchical, network-based, and density-based clustering methods cannot process high-dimensional data. Defining density in a high-dimensional space is particularly tricky with density-based methods. For this reason, the algorithms for description clustering are selected from the partitioning and model-based methods.

Therefore, the **Self-Organizing Map** algorithm by Kohonen [64] is picked a model-based clustering algorithm. Self-Organizing Maps (SOMs) represent a form of artificial neural networks, whose neurons group the data based on unsupervised learning processes. Their functioning is modeled after the processes in human brains, in which delimited areas of neurons are responsible for the stimulus control of some regions of the body.

The x -means and k -medoid algorithms are chosen from the group of partitioning clustering algorithms. The x -means algorithm developed by Pelleg and Moore [93] is based on the calculation rule of the k -means algorithm by Macqueen [82], associating the data to k initially guessed cluster centers and calculating means of the associated data as new cluster centers, cf. Subsection 2.1.2. The method x -means introduces a lower and upper limit for the number of clusters instead of a fixed parameter

Table 4.1: Evaluation of clustering algorithms based on Requirements 4.8 for clustering in the description space

Requirement	SOM	x -means	k -medoids
Homogeneity (i)	✓	✓	✓
Handling High-Dimensional and Noisy Data (ii)	✗	✗	✓
Level of Classification (iii)	✓	✓	✓
Uniqueness (iv)	✓	✓	✓
Outlier Robustness (v)	✗	✗	✗

k and carries out the k -means algorithm for every number within the interval. On the other side, the k -medoids algorithm by Kaufman and Rousseeuw [59] determines the cluster centers by calculating the medoid, see Subsection 2.2.7.

These three algorithms are compared to each other based on the criteria 4.8. For each request made, a checkmark or cross in the third column of Table 4.1 indicates whether this requirement is met or not.

In the following, the evaluation of Table 4.1 is explained in more detail.

All algorithms can meet the requirement of homogeneity. Homogeneity can be achieved by choosing suitable parameters. These are: the network size (SOM), an upper and lower limit of the number of clusters (x -means), or a number of clusters k , which can increase the possibility of minimizing the intracluster similarity.

The k -medoid algorithm is the only algorithm meeting the second criterion of being capable of handling high-dimensional and noisy data. As the SOM and the x -means compute the cluster representative by shifting the data, e.g., averaging all data of one cluster, the noisiness of the data is enhanced. However, a medoid characterizes a set of descriptions by minimizing the average difference to all other descriptions in the cluster. When dealing with noisy real-world data sets, using medoids instead of means as cluster representatives is crucial. Computing the mean over each iteration while clustering would enhance the noisiness of the data leading to distorted clustering results. With the use of medoids, existing descriptions are considered and compared, thus enabling a more precise clustering in the presence of noise.

This is illustrated in the following. Figure 4.6 shows the issue of the calculation of cluster representatives through the example of the FPFH descriptor. The 33 bins of the FPFH algorithm are visualized for each representative, 11 bins for each angle of the descriptor in a different color (θ = red, α = blue, φ = green), dividing the angles into 11 equal intervals. See Subsection 2.1.1 for an explanation of the FPFH algorithm. Here, a real-world wall is sampled with the Velodyne VLP 32-C LiDAR sensor and a synthetic wall as reference is modeled after the same sensor. The mean and the medoid of the real-world wall descriptions are calculated, representing the cluster of a wall-like object e.g., by applying the x -means or the k -medoid algorithms, respectively. In comparison to the synthetic case, where the medoid and the mean description are equal, it can be seen that both real-world representatives are noisy. Additionally, it can be stated that the medoid description is less noisy compared to the mean description. This is due to the fact that the mean calculation averages the noise of every description vector in one vector, while the medoid description contains the noise of only one description vector. Clustering a whole scene enhances this effect, which leads to very noisy mean cluster representatives. With this noisy cluster representative a distinction between different description characteristics is difficult. Thus, the clustering results obtained with algorithms computing the cluster representatives mixing the same entries of multiple descriptions are less precise than without mixing entries.

The third criterion states that over- and underclassification should be avoided. Concerning all clustering algorithms, it depends on the specific parameter. If the grouping of too many or too little clusters is enabled by the parameter selection, over- and underclassification can occur. Therefore, the

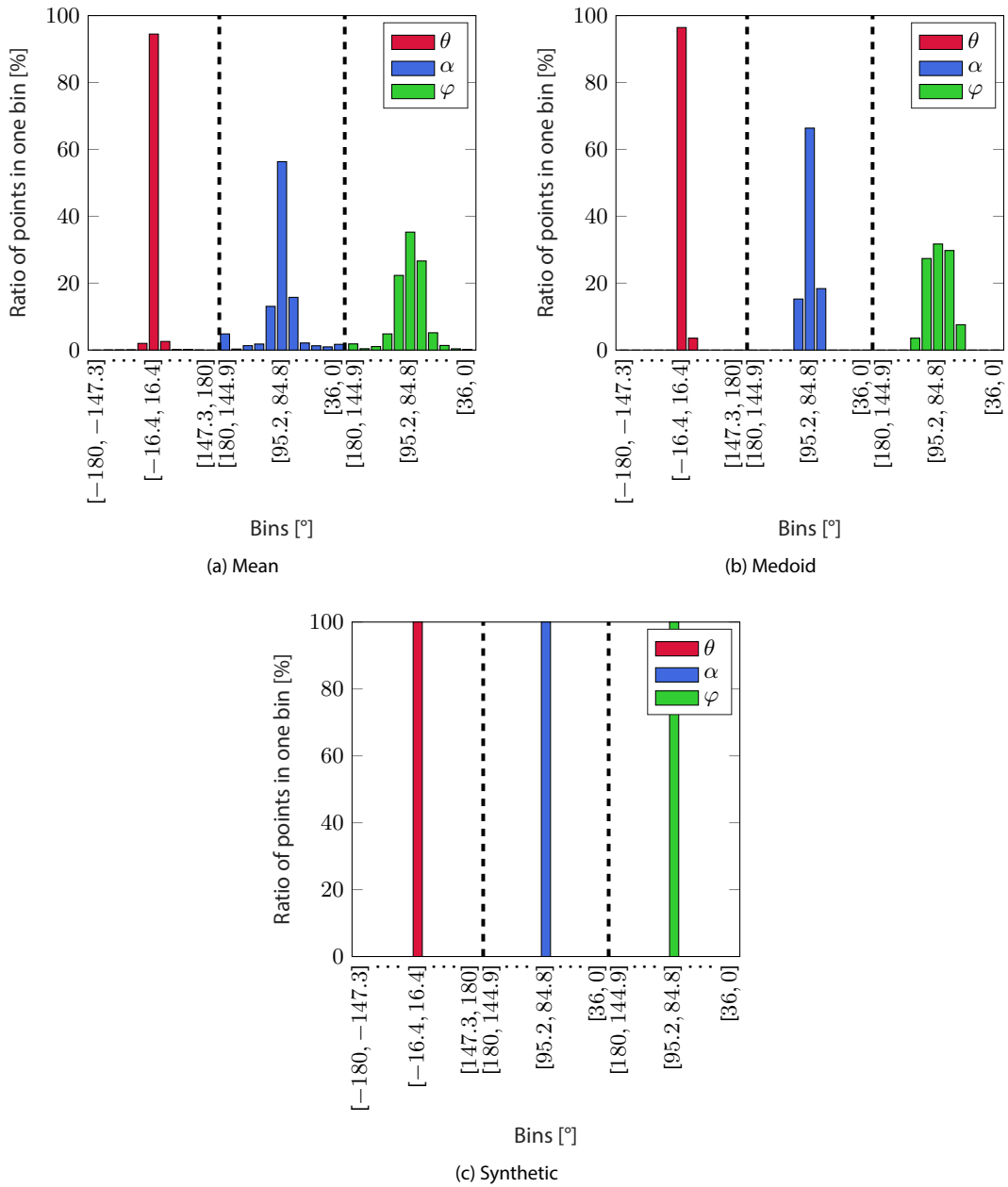


Figure 4.6: Cluster representatives of synthetic and real-world wall sampled with Velodyne VLP 32-C for the FPFH algorithm. Each corresponding FPFH description is plotted, including the first, middle, and last interval limits of each angle.

Table 4.2: Evaluation of clustering algorithms based on Requirements 4.9 for clustering in the Euclidean space

Requirement	DBSCAN	x -means
Uniqueness (i)	✓	✓
Variable Number of Clusters (ii)	✓	-
Outlier Robustness (iii)	✓	✗

parameter choice of each clustering algorithm is crucial, i.e., size of the network topology for the SOM algorithm, upper and lower limit of the number of clusters for the x -means algorithm, and the the number of clusters k for the k -medoid.

Besides, all algorithms fulfill the uniqueness requirements, since the elements of a data set can be clearly assigned to a cluster due to the calculation rules.

According to Fahad et al. [30], all algorithms cannot handle data outliers well, since they are grouped into an existing cluster and are not selected separately. They, therefore, do not meet this requirement. However, the concept presented in Section 4.2 can counter this by sorting out elements that are not statistically meaningful. This is done after the description clustering by keeping the strongly structured clusters, refer to Definition 4.13.

Considering all requirements and especially Requirement (ii), in this thesis, the k -medoids algorithm is chosen as a description clustering algorithm. The preprocessed points of $\mathcal{P}_{\text{preproc map, on-board}}$ are clustered into two classes, resulting in two clusters to avoid over- and underclassification. This is done since geometric information only allows a reliable distinction between curved and non-curved areas, refer to the experiments of the descriptors in Chapter 3, e.g., Table 3.1. For further details about the number of clusters, refer to appendix B.

The description clustering is performed for every selected descriptor of Chapter 3, the FPFH, the SHOT, and the RoPS descriptor. Therefore, the Euclidean distance is used as a metric for the descriptor clustering, as often proposed in the literature [11, 68]. Metrics considering the characteristics of the descriptions, e.g., the cross-bin metric by Cha and Srihari [15], punish differences less hard than the Euclidean distance. Consequently, in this thesis, they are not applied.

This step's outcome are two different groups of non-semantic patterns with small intra-cluster variances compared to the SOM and x -means clustering algorithms.

4.4.2. Implementation of Euclidean Clustering

The second step of the extraction process after the description clustering is to group the two description clusters separately into multiple spatially connected areas. Equivalent to Subsection 4.4.1, in the following, clustering algorithms for Euclidean clustering are compared based on the list of Requirements 4.9.

Many clustering algorithms could be applied for the Euclidean clustering step. In this thesis, two algorithms are chosen which are often applied to this problem. The algorithms DBSCAN by Ester et al. [29], see Subsection 2.2.6, and x -means by Pelleg and Moore [93], see Subsection 2.1.2, are employed. Both algorithms are compared to each other based on the criteria 4.9, whose selection is thereby substantiated. For each request made, a checkmark or a cross in Table 4.2 indicates whether this requirement is met or not. A straight line indicates that the algorithm meets the requirement partially.

In the following, the evaluation of Table 4.2 is explained in more detail.

Both the DBSCAN and the x -means algorithm meet the requirement of a hard clustering process since the elements of a data set can be clearly assigned to a cluster through the calculation rules, cf. Subsection 2.2.6 and Subsection 2.1.2.

The DBSCAN algorithm groups together points of similar density, which can be identified independently of a specified number of clusters or an upper and lower limit. For this reason, the DBSCAN algorithm meets the requirement that a priori no fixed number of clusters must be defined to group the FAs in every type of scene. By defining an upper and lower limit for the number of clusters in the x -means algorithm, a possible range of the actual resulting number of clusters is specified. Its calculation rule enables the statistically best number of clusters to be determined. The x -means algorithm only meets second requirement of enabling a variable number of clusters if a sensible choice of an upper and lower limit for the number of clusters is made.

Density-based algorithms are particularly suitable for the selection of data outliers since they do not have enough points in their neighborhood to form a cluster or to be able to be assigned to an existing cluster. They are explicitly marked as data outliers and are not taken into account for cluster formation. Since the DBSCAN is a density-based algorithm, it is suitable for the selection of data outliers. In contrast, according to Fahad et al. [30], partitioning algorithms, like the x -means, do not deal well with data outliers, since they are grouped into an existing cluster and are not selected separately. Hence, the x -means algorithm does not fulfill the third requirement.

As the DBSCAN fulfills every requirement compared to the x -means algorithm, in this thesis, the DBSCAN is chosen as a clustering algorithm applied for the Euclidean clustering of the 3D point cloud data.

The Euclidean clustering's outcome are multiple groups of a connected set of points with similar description data.

4.4.3. Implementation LIG's Distinctiveness

The distinctiveness is one of the components forming the LIG, refer to Definition 4.14. It is descriptor specific, depending on the characteristics a descriptor captures. In the following, the distinctiveness is defined for each of the selected descriptors, i.e., the PPFH, SHOT, and RoPS.

In the case of the geometry-based **FPFH** descriptor, the distinctness is given, if a bin of the histogram per angle is pronounced and its angle is not scattered. If the calculated angles of the query point's local neighborhood points fall in a common bin, i.e., the description is peaked, the geometry represented by this set of points is distinctive. In contrast, a non-distinctive description shows similarly strong expression in all intervals. For instance, this behavior is expected for vegetation since the normal vectors belonging to these points point in different directions, which leads to strongly varying angle values. Thus, the Sum Of Differences (SOD) is introduced to evaluate the non-uniformity on the basis of the differences between all 11 intervals of the description for each angle θ , α , and φ :

$$\text{SOD}_{\omega}^{\text{FPFH}}(\mathcal{F}) = \sum_{i=1}^{11} \sum_{j=1}^{11} |d_i^{\omega} - d_j^{\omega}| \forall \omega \in \{\theta, \alpha, \varphi\}, \quad (4.7)$$

with d_i^{ω} as the i -th entry of each angle's part of the PPFH description, cf. Subsection 2.1.1. The PPFH description consists of three to $[0, 1]$ normalized histograms for each angle θ , α , φ . For a maximal non-uniform description of an angle, one bin equals 1 the others equal 0, i.e., $\mathbf{d}_{\max}^{\omega} = [1 \ 0 \ \dots \ 0]$. Substituting $\mathbf{d}_{\max}^{\omega}$ in Equation 4.7 yields the maximum SOD of 20 for each angle. For a uniform description of an angle, every bin equals the same, i.e., $\mathbf{d}_{\min}^{\omega} = [\frac{1}{11} \ \dots \ \frac{1}{11}]$. Substituting $\mathbf{d}_{\min}^{\omega}$ in Equation 4.7 yields the minimum SOD of zero for each angle.

In order to obtain a SOD value for an entire PPFH description, the three SOD values are added and normalized:

$$\text{LIG}_{d,\text{FPFH}}(\mathcal{F}) = \frac{1}{60} \cdot (\text{SOD}_{\theta}^{\text{FPFH}}(\mathcal{F}) + \text{SOD}_{\alpha}^{\text{FPFH}}(\mathcal{F}) + \text{SOD}_{\varphi}^{\text{FPFH}}(\mathcal{F})). \quad (4.8)$$

Thus, the $LIG_{d,FPFH}$ is zero for a non-distinctive FPFH description and one for a distinctive FPFH description.

The distinctiveness is calculated similarly for the **SHOT**, since it is also a histogram-based descriptor. The SHOT description is distinctive if the calculated normal differences of the query point's local neighborhood points fall in a common bin, i.e., the description is not uniform. If the calculated normal differences are scattered over every bin uniformly, the description is not distinctive. The Sum Of Differences (SOD) is again used to measure the non-uniformity on the basis of the differences between all 11 intervals of the description for each of the 32 local neighborhood divisions:

$$SOD_{\kappa}^{SHOT}(\mathcal{F}) = \sum_{i=1}^{11} \sum_{j=1}^{11} |d_i^{\kappa} - d_j^{\kappa}| \forall 1 \leq \kappa \leq 32, \quad (4.9)$$

with d_i^{κ} as the i -th entry of each local neighborhood part of the SHOT description, cf. Subsection 2.1.1. The SHOT description consists of 32 to $[0, 1]$ normalized histograms for each part of the local neighborhood. For a maximal non-uniform description of a local neighborhood part, one bin equals 1 the others equal 0, i.e., $d_{\max}^{\kappa} = [1 \ 0 \ \dots \ 0]$. Substituting d_{\max}^{κ} in Equation 4.9 yields again the maximum SOD of 20 for each local neighborhood part. For a uniform description of an angle, every bin equals the same, i.e. $d_{\min}^{\kappa} = [\frac{1}{11} \ \dots \ \frac{1}{11}]$. Substituting d_{\min}^{κ} in Equation 4.9 yields the minimum SOD of zero for each local neighborhood part.

Toward a SOD value for an entire SHOT description, the 32 SOD values are added and normalized:

$$LIG_{d,SHOT} = \frac{1}{640} \cdot \sum_{i=1}^{32} SOD_i^{SHOT}(\mathcal{F}). \quad (4.10)$$

Hence, the $LIG_{d,SHOT}$ is zero for a non-distinctive SHOT description and one for a very distinctive SHOT description.

The **RoPS** descriptor's distinctiveness computation differs from the previous methods, since the individual elements of the 135-dimensional description do not represent the frequency distribution of an individual characteristic but somewhat repetitive values of five independent characteristics. For the RoPS descriptor, the distinctiveness is determined depending on the change in the individual statistics of the 5-tuple, cf. Subsection 2.1.1. Not elongated patterns are particularly important for localization, from which both longitudinal and lateral position information can be derived. These patterns have a comparable statistical point distribution in all rotations or projections. If a statistic value for a selection point changes significantly within the 27 local neighborhood parts of the 5-tuples, there is no similar statistical point distribution in all rotations or projections. In this case, the RoPS description vector is not distinctive. Their standard deviation can describe the change in the individual statistics across all 27 5-tuples so that a value results for each statistic value $(\sigma_{\mu_{11}}, \sigma_{\mu_{12}}, \sigma_{\mu_{21}}, \sigma_{\mu_{22}}, \sigma_e)$. The sum of all standard deviations is build for the distinctiveness:

$$\sigma_{\text{sum}}(\mathcal{F}) = \sigma_{\mu_{11}}^{\eta} + \sigma_{\mu_{12}}^{\eta} + \sigma_{\mu_{21}}^{\eta} + \sigma_{\mu_{22}}^{\eta} + \sigma_e^{\eta} \forall 1 \leq \eta \leq 27. \quad (4.11)$$

A maximum achievable summed standard deviation $\sigma_{\text{sum}} = 0.1$ over all 27 local neighborhood parts follows from the maximum achievable values of the statistics.

This leads to the normalized distinctiveness of the RoPS description:

$$LIG_{d,RoPS}(\mathcal{F}) = 1 - \frac{\sigma_{\text{sum}}(\mathcal{F})}{0.1}. \quad (4.12)$$

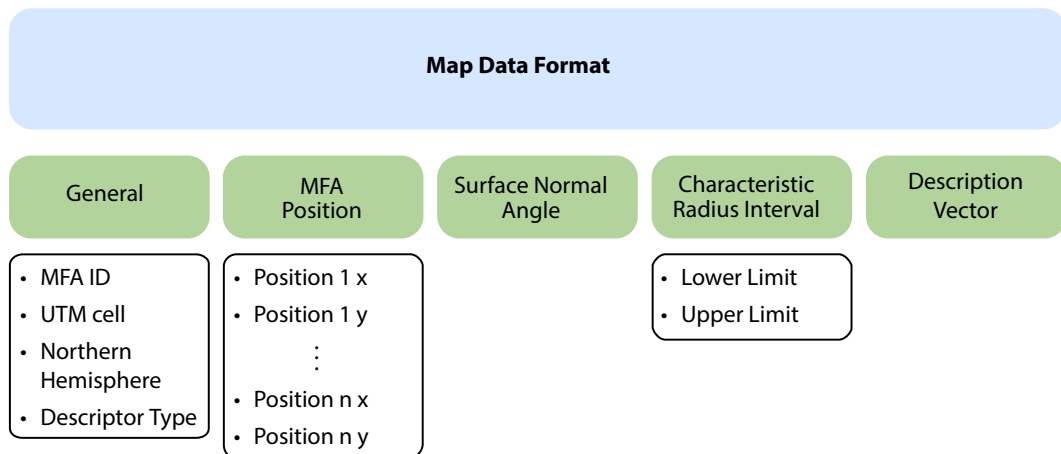


Figure 4.7: Structure and content of the map data format

A distinctiveness of one stands for the continuity of the individual statistics, i.e., a standard deviation of zero, and a distinctiveness of zero stands for the maximum achievable standard deviation.

These definitions are used to compute the LIG's distinctiveness, applied to extract OFAs and MFAs.

4.4.4. Map Data Format

The 2D representations of the map elements, i.e., the MFAs, are represented in a map. This subsection presents the corresponding map data format.

The map contains five blocks of data, i.e., general information, the 2D positions of the MFA representation, the angle of the MFA's surface normal, the lower and upper limit of the characteristic local neighborhood radius, and the description vector, as depicted in Figure 4.7. The *General* data provides a unique identifier for every MFA in an ordered sequence, the MFA's UTM cell, a Boolean value specifying its belonging to the northern hemisphere, and the descriptor type. The identifier is used for its unique assignability.

The specification of the UTM cell and the northern hemisphere's belonging is necessary to provide a uniqueness of the MFA's positions, refer to Subsection 2.2.1. The descriptor type is used to state the descriptor applied to extract the MFAs. The *Feature Area Positions* provide the global coordinates of the 2D representation of the MFAs in the UTM coordinate system. Depending on the shape of the MFA, the number of positions differs. For instance, for a point MFA only one x and one y position is used. From the northing in the mathematically positive direction, the *Surface Normal Angle* is defined as the MFA's orientation. The *Characteristic Radius Interval* denotes the lower and upper limit of the characteristic local neighborhood radius of the MFA, refer to Subsection 3.2.2 and Definition 3.7. The *Description Vector* defines the medoid description, representing the MFA the best. This vector depends on the descriptor type.

4.5. Analyzes of Extraction Process Using Real-World Data

In the following analyzes, the proposed MFA and OFA selection is evaluated on several real-world data sets, checking whether Requirements 4.10 for the extraction method are met. First, position deviations are computed, matching the extracted MFAs with the OFAs, and compared to deviations determined

with a key-point-based extraction method, checking Requirement (i) and (ii). Furthermore, general ideas of the *LiDAR-Feature-based Localization* are evaluated. Therefore, the persistence of the extracted MFAs and OFAs, cf. Definition 4.18, is analyzed with test drives recorded over one and a half years, examining Requirement (iii). Additionally, it is analyzed whether different geometric objects of diverse sceneries are present. The compressibility achieved with the extraction method regarding the map generation is examined comparing the memory size of the raw data and the map data, verifying Requirement (iv).

4.5.1. Position Accuracy Matching Map and On-Board Feature Areas

The first experiment is designed to demonstrate that the extracted MFAs and OFAs are well-suited for determining a 2D global vehicle position by matching sparse on-board data with dense map data, and that the extraction method provides a good tracking stability, cf. Definition 4.17. To begin with, the generation of the map, i.e., MFAs extraction, and the extraction of OFAs are outlined in this context and the method of calculating the global vehicle position, a simple localization algorithm, is proposed. Next, the test environment is introduced, comparing the extraction method of this chapter to another extraction method based on key points. Finally, the position accuracies presented and discussed.

The main idea of localization is to match OFAs with MFAs. This is a map-based localization, and it is applied to estimate the vehicle position with respect to a globally referenced map.

The map is created in three stages. Initially, the dense, globally referenced point cloud \mathcal{P}_{map} is collected by the mobile data acquisition system Trimble MX8 in suburbs and urban areas by repeated scanning of the environment, refer to Subsection 4.1.2 for more details. Next, descriptions are determined for those points of the point cloud $\mathcal{P}_{\text{preproc map}}$ which are suited for the description calculation. For a detailed explanation, refer to Subsection 4.2.1. For that, the descriptors selected in Chapter 3 are taken into consideration, namely, FPFH, RoPS, and SHOT. Last, the MFAs are extracted as defined in sections 4.2, 4.3, and 4.4, building the map. Here, lines instead of polygonal chains are used for the 2D representation. However, this still indicates whether extracted MFAs and OFAs are well-suited for computing a vehicle position.

The OFAs are extracted similarly. First, sparse on-board point clouds $\mathcal{P}_{\text{on-board}}$ sampled by the Velodyne VLP-32C of the same region as the map are the input for the description calculation. The test drive has a total length of approximately 1.5 kilometers. The descriptions are calculated only for suited points of a point cloud $\mathcal{P}_{\text{preproc on-board}}$, refer to Subsection 4.2.1. The same descriptors as for the MFA extraction are applied.

The self-localization of the vehicle with the extracted MFAs and OFAs is realized with a simple algorithm. It is sufficient to show the suitability of the proposed extraction method. However, a well-developed localization algorithm would increase the overall position accuracy but is not necessary to demonstrate the extraction method's capabilities. The implemented localization algorithm consists of three steps. First, the vehicle's pose is initialized from the on-board, close-to-production GPS sensor NovAtel OEMStar [91]. In this way, the approximate position in the map is known and only an excerpt instead of the whole map must be loaded to compare the OFAs with the MFAs. Next, the MFAs and OFAs are the input for the modified ICP algorithm [160], see Subsection 2.2.3, to iteratively match the OFAs with the MFAs considering several rotations and translations. The global 2D vehicle position is finally determined by shifting the initial GPS pose by the rotation and translation computed with the modified ICP.

The 2D positions computed with the extraction method proposed in this thesis are compared to a key-point-based method, namely the ISS [169], refer to Subsection 2.1.2 for details of this method. For this purpose, key points are extracted from the dense globally referenced point cloud \mathcal{P}_{map} , forming the map, and from the sparse on-board point cloud $\mathcal{P}_{\text{on-board}}$. As the ISS outputs many key points

Table 4.3: Average 2D position errors and standard deviations of the errors determined based on the proposed extraction method, considering absolute, longitudinal, and lateral errors in the collected data using a Velodyne VLP-32C. The test drives add up to a total distance of 1.5 kilometers, including suburbs and urban areas [53].

method	absolute [m]	longitudinal [m]	lateral [m]
ISS	1.84 ± 2.8	1.28 ± 1.96	1.5 ± 2.12
proposed w. FPFH	0.54 ± 0.53	0.38 ± 0.41	0.41 ± 0.36
proposed w. RoPS	0.61 ± 0.49	0.47 ± 0.37	0.39 ± 0.36
proposed w. SHOT	0.58 ± 0.59	0.4 ± 0.41	0.43 ± 0.43

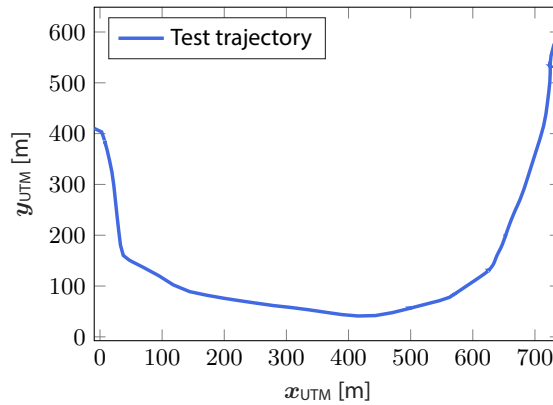


Figure 4.8: Trajectory of the test drives over one and a half years with shifted UTM coordinates, refer to Subsection 3.1.2 [55].

compared to the number of OFAs and MFAs, the modified ICP algorithm would be very slow. Therefore, the extracted key points are used in the unmodified ICP algorithm to calculate a global 2D pose [9], refer to Subsection 2.2.3.

Both global 2D positions are compared to a high-precision GNSS combined with a high-precision IMU, the Applanix POS LV 520 system, serving as a reference [1]. This comparison yields the accuracy.

In Table 4.3, the position accuracies in terms of average RMS trajectory errors are listed for every method. It shows that the proposed algorithm is more precise than the key-point-based ISS extraction method in terms of localization accuracies. This is caused by the fact that key points cannot be detected reliably from LiDAR scan to LiDAR scan and differ a lot from the map detections. Further, the ISS algorithm seems to extract rather random points in on-board point clouds due to its sparsity. Additionally, the results show that the proposed extraction method works with all descriptors, resulting in similar position accuracies.

This experiment shows that the extraction process proposed in this chapter is suitable for localization. Also, it can be obtained from this evaluation that the extraction method is able to generalize independent of the geometry-based descriptor algorithm.

4.5.2. Persistence of Extracted Map and On-Board Feature Areas

In the second experiment, the persistence of the extracted OFAs is analyzed, cf. Definition 4.18. Persistence measures the robustness of the extracted FAs. Robustness is one of the main aspects stating the practicability of the *LiDAR-Feature-based Localization*, cf. research question 1.1 of the introduction. This experiment examines whether the same OFAs matching the MFAs are extracted over a long time

along the same navigational route, see Figure 4.8. This characteristic is a major aspect of the practicality of the *LiDAR-Feature-based Localization*. The extracted FAs need to be robust, i.e., persistent, see research question 1.1.

The MFAs are extracted in three steps. First, the dense, globally referenced point cloud \mathcal{P}_{map} is recorded by the mobile data acquisition system Trimble MX8 by multiple scanning of the environment, see Subsection 4.1.2. Then, for points of the point cloud $\mathcal{P}_{\text{preproc map}}$ suited for the description determination, descriptions are calculated, refer to Subsection 4.2.1. The descriptors selected in Chapter 3 are applied, i.e., FPFH, RoPS and SHOT. Finally, the MFAs are extracted as defined in Section 4.2, Section 4.3, and Section 4.4.

For this analysis, multiple test drives along the same route over one and a half years were performed. Their point clouds $\mathcal{P}_{\text{on-board}}$ of every frame of these test drives sampled with an on-board Velodyne VLP-32C LiDAR sensor are examined. These are used to extract the OFA. The same descriptors as for the MFA extraction are applied to determine the descriptions of these point clouds for suited points of a point cloud $\mathcal{P}_{\text{preproc on-board}}$, refer to Subsection 4.2.1. Since the OFAs are extracted from on-board point clouds, their representation is in the VRF or SRF. They are transformed into the same global coordinate system as the MFAs, i.e., UTM, using an Applanix POS LV 520 reference system for their comparability. All transformed OFAs are accumulated along the entire trajectory and associated with each other by clustering them.

Then, the OFA clusters are matched with the extracted MFAs by comparing their positions. OFA clusters with a position distance of $\lambda_2 = 1$ m and a description distance of $\lambda_1 = 15\%$ to the MFAs' position and descriptions are considered a match, refer to Definition 4.18. The extraction method's persistence is shown by measuring the OFAs' relative frequency as specified in Definition 4.18. This means that OFAs with one matching MFA in every single test drive over one and a half years are considered as 100% persistent. If there is a match with one MFA in only a few test drives the persistence for that OFA decreases accordingly.

This analysis shows that most of the extracted OFAs are persistent and can be detected throughout a long time, in this case, one and a half years. Within the 1.5 kilometer test drive, 83.8% of the selected OFAs are considered sufficiently persistent as they were detected in more than half of our real-world data sets. Besides, even 16.2% of the selected OFAs were detected in each considered data set.

4.5.3. Scenery Coverage

Another main aspect analyzing the practicability of the *LiDAR-Feature-based Localization* is the diversity of sceneries where the FAs are extracted, cf. the research question of the introduction. The motivation of extracting non-semantic elements, i.e., FAs, is to be independent of semantic infrastructure elements. Therefore, this experiment shows that with a geometry-based descriptor many diverse objects can be detected in different sceneries.

In this experiment, dense point clouds of five different sceneries, i.e., urban, rural, suburb, industrial, and on a highway, are used to extract MFAs on the example of the FPFH descriptor. The MFAs were extracted as described in this chapter. Here, 60% of the FAs according to the LIG value are extracted as MFAs.

Figure 4.9 illustrates the three-dimensional view of MFAs detected in these five sceneries. The point of the dense point cloud belonging to MFAs are highlighted in the same color when they belong to the same or MFAs. Points are marked gray if they are not extracted.

The figures show that with the proposed method applying one descriptor various types of objects can be extracted automatically depending on the different geometries present in a scene. In the urban scenery of Figure 4.9c, mostly pillars, curbs, and road signs are extracted. Figure 4.9d shows the detected MFAs of a suburban scenery. Here, mainly house walls and pillars are extracted. In the industrial scenery cf. Figure 4.9e, the house walls of the office blocks and factory buildings and a fence are

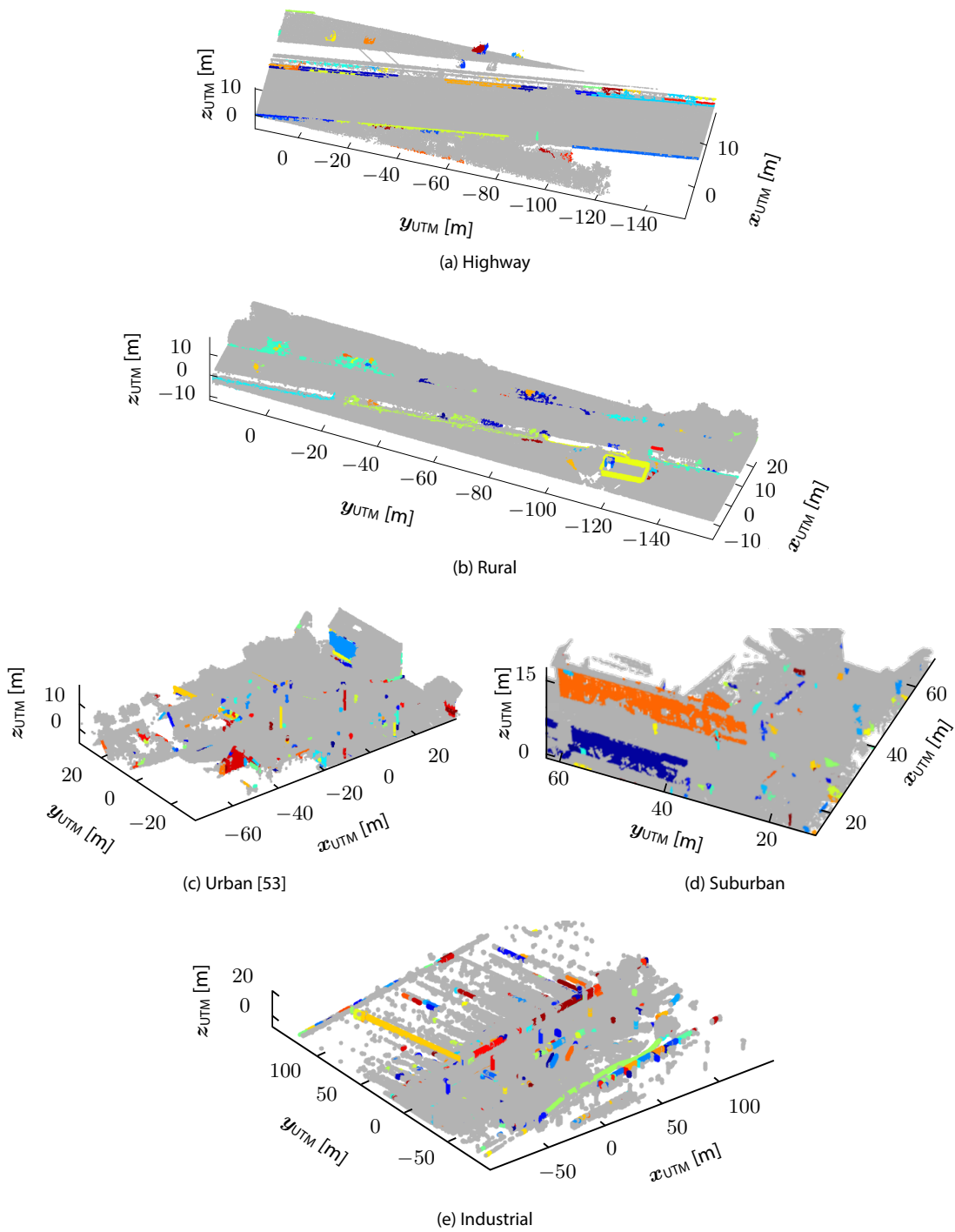


Figure 4.9: Three-dimensional view of sample MFAs (colored areas) in dense point clouds (gray) of five different sceneries using the FPFH descriptor [110]. The dense point clouds are plotted in the shifted UTM coordinate system.

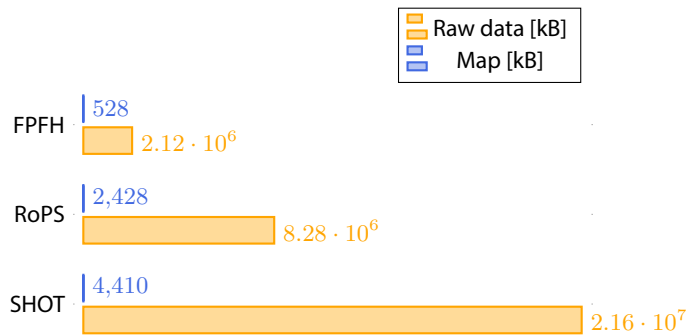


Figure 4.10: Comparison of the memory size of the raw data map point cloud and its descriptions (orange) and the memory size of the map for the FPFH, RoPS, and SHOT descriptor (blue) through the example of a scene.

detected. In these cases, the bushes and trees are discarded as they are not distinctive compared to other objects in each scene. This is different in the rural scenery, see Figure 4.9b. In the rural scenery, the bushes on the side of the road are mostly the only type of object in the scene. Therefore, they are extracted here additionally to a fenced area (yellow) enclosing a shed (blue). On the highway in Figure 4.9a, mostly guardrails are extracted. Additionally, it can be seen that a road sign (light blue) is detected.

However, in all sceneries, additional elements are extracted as the point clouds are not cleaned from dynamic objects, mostly vehicles, which were present during the collection of the dense data. For instance, trucks are extracted in the industrial scenery Figure 4.9e (dark blue, blue, and orange at approximately $(0, 0)$). The description of trucks are very distinctive, similar to walls, and thus are extracted from the point cloud. Further, sometimes clutter is extracted due to boundary effects, e.g., the red element in the right corner in the urban scenery of Figure 4.9c.

Summarizing, this experiment shows that the proposed extraction method is able to detect elements of the environment independently of the type of scenery. Therefore, the proposed approach fulfills the requirement of being independent from infrastructure elements, which was the main motivation of pursuing this research. In a refining step, the map or the dense point cloud should be cleaned from dynamic objects and clutter.

4.5.4. Compressibility of Map Feature Area Extraction

One requirement for using clustering algorithms for the FA extraction is to reduce the memory size of the map, cf. Requirement (iv). This examination analyzes the compressibility of the extraction method by comparing the memory size of the raw data map point cloud and its descriptions with the map's memory size. The map contains all the necessary information of the MFAs, e.g., their ID or 2D point and polygonal chain representation, refer to Subsection 4.4.4 for more details. A simplified investigation is carried out for this purpose, indicating the respective memory size in kilobytes in the comma-separated values file format.

Figure 4.10 shows the results of the analyzes of the memory size depending on the applied descriptor, i.e., FPFH, RoPS, and SHOT, through the example of a suburban area on the test trajectory, cf. Figure 4.8. The memory size for the raw data point cloud and the associated file for the FPFH, RoPS, or SHOT descriptor are marked in orange, for the map file for each descriptor in blue. A memory size of $2.12 \cdot 10^6$ kilobytes can be read off for the sample scene's raw data using the FPFH descriptor. In this case, the compressed map data has a 528 kilobytes memory size, achieving compression factor

of $4 \cdot 10^3$. High compressibility can also be determined using the RoPS and SHOT descriptors. While their descriptions are of higher dimension, the total memory sizes are higher for both the raw and the map data. For the RoPS descriptor, the raw data is compressed from $8.12 \cdot 10^6$ kilobytes to 2428 kilobytes with compression factor of $3 \cdot 10^3$. For the SHOT descriptor, the raw data is compressed from $2.16 \cdot 10^7$ kilobytes to 4410 kilobytes with compression of $5 \cdot 10^3$.

According to the analyzes of the memory size, it can be summarized that, for the FPFH, RoPS, and the SHOT descriptor, a significant reduction in memory size can be achieved using the proposed extraction method compared to saving the entire raw data points with the associated description files. Thus, the Requirement (iv) is fulfilled by the developed extraction algorithm.

4.6. Summary

Whereas Chapter 3 focused on the description of characteristic relations between sampling points in local neighborhoods, the first step of the *LiDAR-Feature-based Localization*, this chapter focuses on the method to extract areas with significant and persistent characteristics automatically, its second step.

For the non-semantic detection of these significant and persistent areas, a set of connected points with similar geometry-based descriptions with a benefit for the localization task is extracted in four steps. First, the geometry-based descriptions are calculated for those points of the point cloud with enough points in their local neighborhood. Then, a clustering algorithm applied in the description space is carried out to group points with similar descriptions using the k -medoid algorithm. These clusters of points with similar descriptions are the input for the clustering in the Euclidean space. Here, the description clusters are grouped separately into multiple areas of spatially connected points using the DBSCAN. In the last step, the best α percent of the resulting clusters are kept, measuring their significance with the benefit for localization, i.e., the Localization Information Gain (LIG), and represented as 2D points and polygonal chains. The LIG unites several values, e.g., measuring the distinctiveness, uniqueness, and spatial diversity of the areas. The outcome of this method are Feature Areas (FAs) in the map, which can be recognized in the vehicle. The map elements are called Map Feature Areas and the detection in the test vehicle are called On-board Feature Areas.

This method is implemented and tested in real-driving scenarios, examining the requirements on the extraction algorithm, cf. Requirements 4.10. For that, a map containing the 2D representations of the Map Feature Areas (MFAs) is generated from a dense and globally referenced LiDAR point cloud, following the proposed algorithm. Real-world point clouds sampled with close-to-production type LiDAR sensors are recorded and used to extract On-board Feature Areas (OFAs). A simple localization algorithm matching OFAs with MFAs is employed to evaluate Requirements (i) and (ii), i.e., the usefulness of the extractions for the localization task. This demonstrates that the proposed extraction algorithm provides persistent and more accurate results than existing key-point-based methods considering global, map-based localization. Verifying the Requirement item (iii), the persistence of the extracted OFAs and MFAs is analyzed by determining the OFAs' relative frequency using data collected over one and a half years along the same navigational route. The findings of this examination demonstrate that most of the extracted OFAs are persistent and can be detected throughout a long time. The compressibility of the proposed algorithm, i.e., Requirement (iv), is analyzed in the third experiment by comparing the memory size of the raw point cloud data and its descriptions with the map's memory size. A significant reduction in the memory size can be accomplished by applying the proposed extraction method. Additionally, it can be seen that the detection of non-semantic FAs enables the possibility to cover more scenes rather than predefined domains.

All of the analyzes demonstrate the potential of a localization relying on non-semantic elements by fulfilling each of the Requirements 4.10. The next chapter presents a robust localization solution using the extracted non-semantic OFAs and MFAs.

Part III.

Positioning with Features

5. Localization using LiDAR Feature Areas

The two preceding chapters presented and analyzed the first two steps of the *LiDAR-Feature-based Localization*. First, descriptor algorithms are applied to determine characterizations for suitable points of on-board and map point clouds sampled with LiDAR sensors. Second, the point cloud information and the descriptions with high information content are used to extract significant and persistent Feature Areas for the map and in the test vehicle automatically. These detections are called Map Feature Areas (MFAs) and On-board Feature Areas (OFAs), respectively. In this chapter, these non-semantic extractions are integrated into a robust localization algorithm, evaluating the practical relevance of the approach of the *LiDAR-Feature-based Localization*. The localization forms the third step of the *LiDAR-Feature-based Localization*. Therefore, the methodology of applying a localization algorithm is outlined, including the criteria for a robust localization algorithm and the research questions of this chapter, see Section 5.1. The choice of a localization algorithm, operating on the FAs, is made based on these criteria. The integration of the FAs into the localization algorithm is presented in Section 5.2. The practical relevance of the FAs' application to a robust localization is analyzed and its results are presented in Section 5.3. These experiments evaluate several aspects of practical relevance of the *LiDAR-Feature-based Localization*: position accuracies in Subsection 5.3.1, and the necessary computing resources in Subsection 5.3.2.

5.1. Terms and Methodology

Here, the methodology of the localization algorithm is provided. It includes criteria for choosing a localization algorithm integrating non-semantic elements. It also contains the research questions addressed in this chapter.

5.1.1. Defining the Localization

The terms of the third step of the *LiDAR-Feature-based Localization* are specified in the following subsection. The third step is to match the OFAs with the MFAs computed with the same descriptor in a robust way to achieve an accurate 2D pose of the vehicle.

The vehicle pose must be defined, which is the output of the localization algorithm.

Definition 5.1 (Vehicle Pose). Let $x_v \in \mathbb{R}$ be the easting position, let $y_v \in \mathbb{R}$ be the northing position, and let $\theta_v \in [0, 2\pi)$ be the vehicle's heading, every scalar as defined in the UTM coordinate system. Then, the vehicle pose is defined as $\chi_v = [x_v, y_v, \theta_v]$.

The vehicle pose is specified in a global coordinate system, here the UTM reference frame. The application of the standardized global Cartesian coordinate system has two reasons.

Since Cartesian systems enable distance computations with low distortions, they are easier to deal with when processing data, for example, compared to an ellipsoid coordinate system like the WGS84.

Additionally, the vehicle pose is required in a global coordinate system. For instance, V2X systems send and receive messages, which include their poses, to enhance road safety, e.g., with special vehicle warnings. The vehicle pose must be given in a global coordinate system so that the receivers are capable of interpreting the messages' contents. At the same time, the vehicle pose is needed in

a map-relative system. For example, the path planning task depends on a priori knowledge, i.e., map data, since they require further information that the vehicle might not be able to derive from its on-board sensor data. This could be the case if road markings bounding the vehicle's lane are not precisely recognized in unfavorable weather conditions. Hence, such tasks also depend on the vehicle's pose within a given map to be able to obtain map information. Therefore, the map elements are also defined in the UTM reference system, refer to Subsection 4.4.4.

Since this thesis focuses on two-dimensional localization for simplification, the vehicle pose is defined in the two-dimensional UTM coordinate system.

Thus, the goal of the third step of the *LiDAR-Feature-based Localization* can be formulated as follows.

Problem Formulation:

Let $\mathcal{F}_O \subset \mathcal{P}_{\text{on-board}}^{x,y,z} \times \mathcal{D}_{\mathcal{P}_{\text{on-board}}}$ be a set of On-board Feature Areas of different times and let $\mathcal{F}_M \subset \mathcal{P}_{\text{map}}^{x,y,z} \times \mathcal{D}_{\mathcal{P}_{\text{map}}}$ be a set of Map Feature Areas computed with the same descriptor. The goal is to estimate the most likely two-dimensional vehicle pose $\chi_v \in \mathbb{R}^2 \times [0, 2\pi)$ depending on all OFAs matching the MFAs.

5.1.2. Procedure of Localization with LiDAR Feature Areas

The procedure of approaching the localization algorithm consists of two steps: the integration of FAs into a selected localization method, and the evaluation of the proposed algorithm in the context of *LiDAR-Feature-based Localization*. In the following, the methodology of both steps is explained in detail.

Selecting and Expanding the Localization Algorithm: This thesis' approach of a localization relying on non-semantic elements should be revised in this chapter. Therefore, the approach should be examined based on a state-of-the-art localization algorithm, applying the extracted FAs. In the following, several requirements are defined, which the localization algorithm has to fulfill.

Requirements 5.2 (Localization Algorithm). A localization algorithm estimating the most likely vehicle pose relying on On-board Feature Areas \mathcal{F}_O and matching Map Feature Areas \mathcal{F}_M , has to fulfill the following conditions:

- (i) *Full SLAM:* The localization algorithm has to enable the consideration of past vehicle poses.
- (ii) *Small Influence of Linearization Errors:* The localization algorithm has to mitigate the influence of linearization errors on the estimation.
- (iii) *Vehicle Applicability:* The localization algorithm has to be applicable on-board in a vehicle environment.

In the following, the requirements listed above for selecting of a localization algorithm are described in more detail.

The first requirement states that a localization algorithm should factor in the information of past times, not only the current observations. This includes past vehicle poses and OFAs extracted in previous point clouds. The requirements state that past determined vehicle poses should be updated, making use of the current information. That means that the *full SLAM* problem should be solved. This requirement enables the localization algorithm to make the vehicle pose determination more robust by applying past information.

Table 5.1: Evaluation of localization algorithms based on Requirements 5.2 for LiDAR-Feature-based Localization

Requirement	Filtering-based	Smoothing-based
Full SLAM (i)	✗	✓
Small Influence of Linearization Errors (ii)	✗	✓
Vehicle Applicability (iii)	✓	-

Since, in practice, the localization problem is often non-linear, its linearization is carried out for efficiency reasons, incorporating linearization errors. Therefore, in the second requirement, it is defined that the localization algorithm should keep linearization errors as small as possible.

The third requirement states that a localization algorithm should be applicable in an on-board test vehicle. That means that it should provide accurate and robust results. Additionally, the method's computational effort should fit to on-board environments.

Hereafter, these requirements are used to select an appropriate localization algorithm, refer to Table 5.1 for a summary. For each criterion, a checkmark or a cross in Table 5.1 displays whether this requirement is fulfilled or not. A straight line displays that the method fulfills the requirement partially. In the literature, localization approaches are divided into two categories. Such methods can be defined as filtering- or smoothing-based approaches. Further, in recent years, AI-based approaches have been developed, approaching the localization problem. However, AI-based approaches are not taken into account in this thesis.

Filtering-based methods formulate the localization problem as estimating the vehicle's pose at one timestamp with the current measurements and the map. Popular methods belonging to this category are the two-stepped Extended Kalman Filter (EKF), particle, or information filters [131]. Thus, Requirement 5.2 (i) is not met. They assume the Markov property and thus, large linearization errors can occur, contradicting Requirement 5.2 (ii) [85]. However, just solving the localization problem for the current measurements induces a low computational effort.

Smoothing-based methods view the localization problem as solving a sequence of vehicle poses at specific time instances, meeting Requirement 5.2 (i). Usually, these methods depend on least-square error minimization strategies. These algorithms are able to reduce the influence of linearization errors by applying relinearization repeatedly. Therefore, they fulfill Requirement 5.2 (ii). One downside to these methods is that the problem which has to be solved expands over time. Hence, the computational effort grows and reduces the vehicle applicability, i.e., Requirement 5.2 (iii). Several procedures have been developed to overcome this effect, including a sliding window. This sliding window either removes old measurements and adds new ones or performs marginalization. Marginalization is the strategy preserving old measurements' data within the optimization problem while deleting them from the estimation.

In this thesis, taking all requirements into account, a smoothing-based approach is selected as a localization algorithm. An intuitive way of expressing the smoothing-based methodology is by the graph-based framework utilized in this thesis. This approach can be viewed constructing a factor graph, refer to Subsection 2.2.8 for more details. Nodes in the factor graph are current and previous poses of the vehicle, OFAs, and MFAs. Edges between two nodes in the factor graph are measurements limiting the nodes. The localization algorithm works by computing a factor graph which explains best the measurements. In this thesis, a sliding window approach is applied to overcome the growing computational effort. For more details, refer to Subsection 2.2.8.

As this is a state-of-the-art process, the main task of the implementation is to integrate the OFAs and MFAs into the factor graph-based localization algorithm. In this thesis, the work of Wilbers et al. [160] provides a basis for the localization algorithm.

Evaluating the Localization Algorithm: The overall concept of applying the localization algorithm should be evaluated for their application in the real world. Consequently, requirements for evaluating the *LiDAR-Feature-based Localization* are stated below, referring to the research question defined for this thesis in Section 1.1. This thesis focuses on the potential of applying non-semantic information to localization. Therefore, the requirements do not include the criteria like real-time capability and the computational effort.

Requirements 5.3 (Localizing with On-Board and Map Feature Areas). *A localization algorithm, matching OFAs and MFAs, has to fulfill the following conditions:*

- (i) *Accuracy: The LiDAR-Feature-based Localization has to provide accurate results.*
- (ii) *Long-Term Robustness: The LiDAR-Feature-based Localization has to provide accurate results over a long time, e.g. a year.*
- (iii) *Scene Coverage: The LiDAR-Feature-based Localization has to provide accurate results in multiple types of scenes, e.g., suburbs, rural areas, and urban areas.*

The preceding criteria for a localization relying on non-semantic data are described in more detail in the following.

In the first requirement, it is defined that the non-semantic localization should provide accurate results. Since localization is developed in the context of automated driving, it should meet the accuracy demands of this application. Reid et al. [104] derived accuracy requirements for an US freeway and local roads. They state that a passenger vehicle on an US freeway should achieve a 95% accuracy in lateral direction of 0.14 meters and in longitudinal direction of 0.48 meters. On US local roads, a passenger vehicle should achieve a 95% accuracy in lateral and longitudinal direction of 0.10 meters. These specifications are used in this chapter to evaluate the localization results.

The second requirement specifies that the non-semantic localization can be applied over a long time, e.g., one year. As the semantics of the OFAs and MFAs are not known, their persistence cannot be assumed. Therefore, the long-term robustness is crucial to ensure a localization that can be applied during all seasons of the year.

In the last requirement, it is stated that the non-semantic localization shall extract FAs which provide accurate localization in various types of scenes, like suburbs, rural areas, and urban areas. As the motivation of the *LiDAR-Feature-based Localization* is to be independent of infrastructure elements, this requirement embraces the core criterion. That is why its fulfillment is crucial.

These criteria are used to evaluate the practical benefit of the *LiDAR-Feature-based Localization*.

Sensors: For the calculation of the vehicle pose, not only the OFAs and MFAs, extracted from on-board and offline generated LiDAR scans are used, but also the vehicle's odometry data, and GNSS data. Details of the sensors recording these data are summarized hereafter. A detailed description about the sensors and the architecture of the used test vehicles can be found in Subsection 3.1.2.

The close-to-production LiDAR sensors Velodyne VLP 16 [151] and Velodyne VLP 32-C [150] are used to extract OFAs. The LiDAR sensors sample the environment non-uniformly with a higher horizontal resolution compared to the vertical resolution. The resulting scans are sparse and afflicted with noise. Furthermore, the sensors' sampling density decreases with increasing distance between the sensor and the scanned object. As a result, the point clouds are only an incomplete representation of the real-world.

The Trimble MX8 LiDAR sensor integrated into a mobile data acquisition system is used to extract MFAs [105, 142]. This sensor measures the surroundings with high precision by repeated scanning and postprocessing. This process yields a globally referenced dense point cloud with a sampling resolution of approximately $34 \frac{\text{points}}{100 \text{ cm}^2}$. The point clouds obtained represent a complete and ideal image of the real world.

A software module called EgoMaster [3] estimates the odometry data based on data from standard and close-to-production type sensors integrated into the ESP and ABS. It provides data with 100 Hertz.

GNSS data is received with standard production-type GPS sensors. In this thesis, the NovAtel FlexPak-G2 OEMStar [91] is used. This is a close-to-production type GPS sensor.

5.1.3. Research Questions of the Application of the Localization Algorithm

The research questions addressed in this chapter are introduced and depicted in detail in the following subsection.

How should FAs be integrated into a state-of-the-art localization algorithm?

- Most graph-based localization algorithms rely on semantic landmarks like pillars, house buildings, or road markings. However, in this thesis, non-semantic data needs to be integrated into a graph-based localization method. As these on-board and map elements are composed of 2D position information, i.e., points and polygonal chains, and descriptors, their localization information needs to be integrated suitably.
- Section 5.2 presents the integration of the OFAs and MFAs into the graph-based localization.

What is the practical benefit of the LiDAR-Feature-based Localization?

- Compared to the sensing of semantic data, the *LiDAR-Feature-based Localization* detection of non-semantic data is more expensive and time-consuming, assuming no prior knowledge about the LiDAR data. This additional effort is tolerable if this induces independence of infrastructure elements for localization. Therefore, the practical benefit must be examined.
- Section 5.3 presents the results of several examinations of the practical benefit of the *LiDAR-Feature-based Localization*.

The publication [55] arose from this thesis and presents the localization method's integration of FAs. It depicts the localization algorithm and accuracy results when integrating point and line FARs.

5.2. Integration of Features into Graph-Based SLAM

In the following section, the integration of the OFAs and MFAs into the localization algorithm is presented. In the following, the OFAs and MFAs are considered for each descriptor type separately. An overview of one cycle of the localization algorithm, i.e., one time step, is illustrated in Figure 5.1. The algorithm's main idea is a sliding window graph-based localization with non-semantic LiDAR FAs on a priori generated map. 1) *Collection of OFAs* The OFAs are extracted from on-board LiDAR scans $\mathcal{P}_{\text{on-board}}$ calculating descriptions with a descriptor algorithm.

2) *Association of FAs* They are associated over time regarding OFAs from consecutive timestamps corresponding to each other. The associated OFAs are matched with MFAs. Subsequent, the associated OFAs, the matched MFAs, the vehicle's odometry and GNSS measurements are the input for a factor graph generation.

3) *Graph-based Optimization* The factor graph is optimized to estimate the vehicle pose. These three steps, carried out at predefined time steps, are explained in more detail, focusing on the last two steps in Subsection 5.2.1, Subsection 5.2.2, Subsection 5.2.3. Additionally, the map generation process, i.e., the detection of MFAs is summarized in Subsection 5.2.1. A sliding window is integrated into the optimization problem to reduce the computational effort. The principle of the sliding window is explained in Subsection 5.2.6.

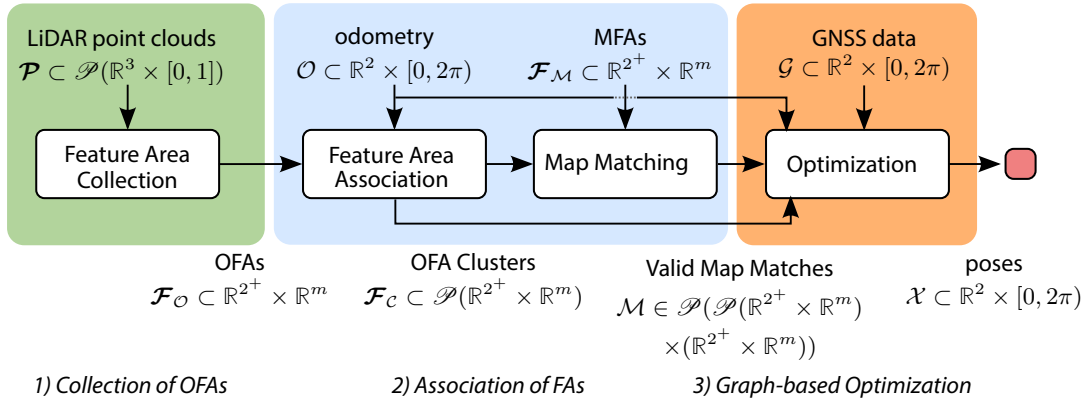


Figure 5.1: Scheme of one time step of the localization computation using LiDAR point clouds, GNSS data, odometry, and map information. 1) For every timestamp, OFAs are extracted from point clouds through descriptor calculations (green). 2) These OFAs are clustered, matched with MFAs, and stabilized, taking previous map matches into account (blue). 3) The vehicle pose is estimated building and optimizing a factor graph from all input data (orange) [55].

5.2.1. Collection of On-Board Feature Areas and Map Generation

At first, the OFAs are extracted, as described in Chapter 4. A summary is given below.

Descriptions of suitable points of a LiDAR scan $\mathcal{P}_{\text{preproc on-board}} \subset \mathcal{P}_{\text{on-board}}$ are calculated performing a descriptor algorithm. These descriptions are determined for those points whose local neighborhood includes enough points to characterize a three-dimensional structure.

Since descriptions are computed for nearly every point of the point cloud, the data size should be reduced and only those data should be used for localization which is robust and useful for localization. Therefore, the next two steps consist of a two-stepped clustering method, compressing the 3D point cloud and description data. Both the descriptions and the points' euclidean coordinates are taken into account. First, the data is compressed by clustering points with similar descriptions while preserving most of the useful information. For this step, the k -medoids algorithm by Kaufman and Rousseeuw [59] is applied. In Subsection 4.4.1, a detailed reason of the choice of the cluster algorithm is given. In doing so, the descriptions $\mathcal{D}_{\mathcal{P}_{\text{preproc on-board}}}$ are sorted into two classes because noisy descriptions only permit a reliable distinction between two classes. Taking the example of the FPFH descriptor, this yields a distinction between strongly curved and weakly curved descriptions. Second, the two description cluster are grouped separately into multiple spatially connected areas. This is realized with the DBSCAN algorithm by Ester et al. [29]. As a result, this process detects multiple areas of spatially connected points with similar descriptions, i.e., FAs.

The detected FAs are modeled as 2D points and polygonal chains. Since this thesis concentrates on 2D localization, a 2D representation is sufficient. A FA with a local expansion in x - and y -direction, which is smaller than 0.5 meters, is represented as a point. If that condition is not fulfilled, the FA is represented as a polygonal chain. Because elongated objects are often formed like the course of the road in the real world, e.g., walls or curbs, they are not always linear. Therefore the FAs are modeled as polygonal chains, not lines.

Next, those FAs are extracted from the set of FAs which are useful for localization. For this purpose, a set of different measures describing the FAs' suitability is used. They are summarized in the Localization Information Gains (LIGs). For on-board point clouds, it captures the distinctiveness, i.e., how explicitly a FA's can be described, the uniqueness, i.e., the identifiability within a point cloud by the FA's description, and the spatial diversity, i.e., the contribution of the FA to the spatial distribution within a point cloud, refer to definitions 4.14, 4.15, 4.16. One LIG value, containing the three components, is

computed for every FA of the point cloud. The FAs with the highest LIG value are extracted as OFAs.

The process is repeated for every point cloud within each time step. This yields OFAs $\mathcal{F}_{\mathcal{O},t_i}$ at times $t_i \in \mathbb{R}$.

This is done similarly for the offline MFAs extraction. Instead of sparse on-board LiDAR point clouds dense, globally referenced point clouds are used. The clustering is carried out in the same way. However, only the distinctiveness is considered while extracting MFAs from the set of FAs. This is due to the fact that only those FAs with good descriptor characteristics should be kept. The LIG values' spatial diversity and uniqueness cannot be computed based on the single FAs as they are point-cloud dependent.

5.2.2. Association of Feature Areas

The OFAs are extracted for every point cloud for every time step for each descriptor type separately, as described in the previous subsection. In the second step, an association of OFAs is carried out. This is done to determine which OFAs, extracted from the on-board point clouds of the current time step, can be matched with OFAs extracted from point clouds of former time steps.

The idea of association is to compare the current OFAs with previously associated OFAs determined with the same descriptor, or rather their representatives. All associated OFAs are grouped into an OFA clusters, whose definition is given below. Only same types of OFAs are clustered, i.e., computed with the same descriptor and with the same position representation (points or polygonal chains).

Definition 5.4 (On-Board Feature Area Cluster). *Let $t_1, \dots, t_n \in \mathbb{R}$ be the points in time of consecutive on-board measurements, let $\mathcal{F}_{\mathcal{O},t_i} \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be an On-board Feature Areas at time $t_i \in \mathbb{R}$ with either point or polygonal chain position representations $\mathbf{r}_{t_i} \in \mathbb{R}^{2^+}$ and description representatives $\mathbf{d}_{t_i} \in \mathbb{R}^m$ determined with a descriptor function, let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some descriptor specific metric with metric threshold $\varepsilon_1 \in \mathbb{R}_{>0}$, and let $m_p : \mathbb{R}^{2^+} \times \mathbb{R}^{2^+} \rightarrow \mathbb{R}_{\geq 0}$ be some metric with metric threshold $\varepsilon_2 \in \mathbb{R}_{>0}$. Then, an On-board Feature Area Cluster (OFA cluster) is defined by the set $\mathcal{F}_{\mathcal{C},\{t_1,t_n\}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$, containing all On-board Feature Areas whose representations and descriptions fulfill the following conditions*

- (i) $m_p(\mathbf{r}_{t_i}, \mathbf{r}_{t_j}) < \varepsilon_1 \forall i, j = 1, \dots, n$,
- (ii) $m_d(\mathbf{d}_{t_i}, \mathbf{d}_{t_j}) < \varepsilon_2 \forall i, j = 1, \dots, n$.

The position and description data is compared to find those OFAs which are likely the same. Here, a distinction is made between two types of OFAs, considering the 2D representation of the OFAs. Point OFAs clusters can only consist of point OFAs, and polygonal chain OFAs clusters can only consist of polygonal chain OFAs. As position and description data is compared, a weighting can be performed with ε_1 and ε_2 .

In this thesis, the following metrics are applied. For the descriptor specific metric m_d , (normalized) Euclidean distances like Equation 3.2, 3.3, and 3.4 are used. For the metric m_p measuring the spatial differences, a distinction is made between points and polygonal chains, containing at least two points. Considering points, the Euclidean distance is used. Considering polygonal chains containing at least two points, the metric used is more complex. For every point of an OFA cluster representative $\mathbf{r}_p \in \mathbb{R}^2$, its orthogonal projection $\mathbf{o} \in \mathbb{R}^2$ onto every MFA line segment of the polygonal chain, i.e., with start point $\mathbf{s}_1 \in \mathbb{R}^2$ and endpoint $\mathbf{s}_2 \in \mathbb{R}^2$, is determined:

$$\mathbf{o} = \mathbf{s}_1 + \underbrace{\frac{(\mathbf{r}_p - \mathbf{s}_1)^\top (\mathbf{s}_2 - \mathbf{s}_1)}{(\mathbf{s}_2 - \mathbf{s}_1)^\top (\mathbf{s}_2 - \mathbf{s}_1)}}_{=:t} (\mathbf{s}_2 - \mathbf{s}_1) \text{ if } 0 \leq t \leq 1. \quad (5.1)$$

Then, for every orthogonal projection, the minimal Euclidean distance of o to the nearest point of the polygonal chain of the MFA is computed. The mean of all minimal Euclidean distances yields m_p .

Next, the position representative of a set of polygonal chains and set of points must be defined.

Definition 5.5 (Position Representative of Set of Points). Let $t_1, \dots, t_n \in \mathbb{R}$ be the points in time of consecutive on-board measurements and let $\mathbf{r}_{t_1} = [r_{t_1,1} \ r_{t_1,2}]$, \dots , $\mathbf{r}_{t_n} = [r_{t_n,1} \ r_{t_n,2}] \in \mathbb{R}^2$ be point position representations of a OFA cluster. Then, a position representative $\mathbf{r}_{\{t_1, t_n\}} \in \mathbb{R}^2$ is defined as

$$\mathbf{r}_{\{t_1, t_n\}} = \left[\frac{1}{n} \sum_{i=1}^n r_{t_i,1} \quad \frac{1}{n} \sum_{i=1}^n r_{t_i,2} \right].$$

Definition 5.6 (Position Representative of Set of Polygonal Chains). Let $t_1, \dots, t_n \in \mathbb{R}$ be the points in time of consecutive on-board measurements and let $\mathbf{r}_{t_1}, \dots, \mathbf{r}_{t_n} \in \mathbb{R}^{2^+}$ be polygonal chain position representations of a OFA cluster. Then, a position representative $\mathbf{r}_{\{t_1, t_n\}} \in \mathbb{R}^{2^+}$ is determined from $\mathbf{r}_{t_1}, \dots, \mathbf{r}_{t_n}$ with the Ramer-Douglas-Peucker algorithm [27, 103].

Note, that details about the Ramer-Douglas-Peucker algorithm can be found in Subsection 2.2.4.

Then, the OFA cluster representative for the current time step can be defined.

Definition 5.7 (On-Board Feature Area Cluster Representative). Let $\mathcal{F}_{C, \{t_1, t_n\}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ be an On-board Feature Area cluster of times $t_1, \dots, t_n \in \mathbb{R}$, and let $\mathbf{r}_{\{t_1, t_n\}} \in \mathbb{R}^{2^+}$ be the position representative and $\mathbf{d}_{\{t_1, t_n\}} \in \mathbb{R}^m$ the description representative, i.e., medoid description, of all elements in $\mathcal{F}_{C, \{t_1, t_n\}}$. Then, an On-board Feature Area cluster representative (OFA cluster representative) $\mathcal{C}_{\{t_1, t_n\}} \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ of times $t_1, \dots, t_n \in \mathbb{R}$ of the On-board Feature Area $\mathcal{F}_{C, \{t_1, t_n\}}$ is defined by

$$\mathcal{C}_{\{t_1, t_n\}} = (\mathbf{r}_{\{t_1, t_n\}}, \mathbf{d}_{\{t_1, t_n\}}).$$

The OFA cluster representative consists of one description and the spatial information, i.e., a point or a polygonal chain. In this thesis, the description is calculated as the medoid description of all OFAs descriptions.

The OFA association consists of four steps. All of them are explained in the following. Note that only point OFAs can be associated with point OFAs, and polygonal chain OFAs with polygonal chain OFAs. In this thesis, OFAs determined with the same descriptor function are associated. Initially, every OFAs extracted from the first point cloud of a test drive are made into singleton clusters each portrayed with a OFAs cluster representative. The association process is carried out for all subsequent point clouds or rather OFAs cluster representatives based on the local vehicle's odometry data \mathcal{O} . OFA cluster representatives from previous time steps are translated and rotated, interpolating the odometry data to the time of each current OFAs, i.e., the average time of all points of their corresponding FA. The translated and rotated OFA cluster representatives and current OFAs cluster representative of the point cloud are clustered checking conditions (i) and (ii) of Definition 5.4. That means if the distance of the description of a currently detected OFA to an OFA cluster representative's description is small enough and if the distance of the current OFAs' position to an OFA cluster representative's position is small enough, the OFA cluster is updated. That means, a current OFA is added to that OFA cluster and new OFA cluster representatives are determined from all elements within the cluster. If one of the two conditions is not met, a new cluster is created, containing only this current OFA.

Thus, the association process groups similar OFAs into clusters. This enables the association of data from several time steps and point clouds. At the same time, arising position errors of the OFAs are averaged by calculating the position of the OFA cluster representative from all elements of that cluster, assuming uniform distributed noise of the sensor data.

5.2.3. Map Matching

After the association of OFAs, it is decided which of the OFA clusters, the output of the previous step, correspond to which MFAs. In this thesis, this is called map matching. The map matching process is performed once in every time step.

In the case of one OFA cluster corresponding to one MFA, the pair is called a map match. This is defined in the following.

Definition 5.8 (Map Match). Let $\mathcal{C}_{\{t_1, t_n\}} = (\mathbf{r}_{\{t_1, t_n\}}, \mathbf{d}_{\{t_1, t_n\}}) \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be an On-board Feature Area cluster representative transformed into the UTM system with either a point or polygonal chain position representation of an On-board Feature Area cluster $\mathcal{F}_{\mathcal{C}, \{t_1, t_n\}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ of times $t_1, \dots, t_n \in \mathbb{R}$ determined with a descriptor function, let $\mathcal{F}_{\mathcal{M}} = (\mathbf{r}_{\mathcal{M}}, \mathbf{d}_{\mathcal{M}}) \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be a Map Feature Area, let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be a descriptor specific metric with metric threshold $\varepsilon_1 \in \mathbb{R}_{>0}$, and let $m_p : \mathbb{R}^{2^+} \times \mathbb{R}^{2^+} \rightarrow \mathbb{R}_{\geq 0}$ be a metric with metric threshold $\varepsilon_2 \in \mathbb{R}_{>0}$. Then, the 2-tuple $(\mathcal{F}_{\mathcal{C}, \{t_1, t_n\}}, \mathcal{F}_{\mathcal{M}}) \in \mathcal{P}(\mathbb{R}^{2^+} \times \mathbb{R}^m) \times (\mathbb{R}^{2^+} \times \mathbb{R}^m)$ is called map match if the following conditions are fulfilled

$$(i) \quad m_d(\mathbf{d}_{\{t_1, t_n\}}, \mathbf{d}_{\mathcal{M}}) < \varepsilon_1$$

$$(ii) \quad m_p(\mathbf{r}_{\{t_1, t_n\}}, \mathbf{r}_{\mathcal{M}}) < \varepsilon_2.$$

A map match is only defined for either point OFAs and point MFAs, or polygonal chain OFAs and polygonal chain MFAs. The metrics used in this thesis for the map matching are the same as for the association of FAs, refer to Subsection 5.2.2.

Map matching includes three steps. Note that only point OFAs can be matched with point MFAs, and polygonal chain OFAs with polygonal chain MFAs. In this thesis, OFAs are matched to MFAs which were determined with the same descriptor function.

As the MFAs are specified in the global UTM reference system and the OFA cluster representatives in the VRF of the current time, the OFAs are transformed into the UTM system first. The conversion is carried out by transforming all OFAs cluster representatives into the UTM system with a preceding determined vehicle pose. It is accurate enough to use no current, but an outdated pose since the global positions are just processed to limit the search space to match the OFAs with the MFAs. In this way, not the whole map needs to be searched.

Second, to find OFA cluster representative matching the MFAs, the best rotation and translation between them are determined. For this determination, the modified ICP is applied, refer to Subsection 2.2.3. The application of the modified ICP in the context of FAs is presented hereafter. The OFA cluster representatives are iteratively rotated and translated. Therefore, the following steps are performed for each rotation and translation. The Euclidean distances between the positions of each point OFA cluster representative and point MFA is computed, assuming at least one point OFA exists. For each Euclidean distance which is small enough, every OFA cluster representative is shifted with the Euclidean distance. The best shift of all Euclidean distances is computed by taking not only the shifted OFAs cluster representative's and MFAs' position but also their description into account. Here, a distance measures the similarity of one OFA cluster representative with one MFA. This distance is called a map matching error and is defined as follows.

Definition 5.9 (Map Matching Error). Let $\mathcal{C}_{\{t_1, t_n\}} = (\mathbf{r}_{\{t_1, t_n\}}, \mathbf{d}_{\{t_1, t_n\}}) \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be an On-board Feature Area cluster representative transformed into the UTM system with either a point or polygonal chain position representation of an On-board Feature Area cluster $\mathcal{F}_{\mathcal{C}, \{t_1, t_n\}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ of times $t_1, \dots, t_n \in \mathbb{R}$ determined with a descriptor function, let $\mathcal{F}_{\mathcal{M}} = (\mathbf{r}_{\mathcal{M}}, \mathbf{d}_{\mathcal{M}}) \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be a Map Feature Area, let $m_d : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}_{\geq 0}$ be some descriptor specific metric, and let $m_p : \mathbb{R}^{2^+} \times \mathbb{R}^{2^+} \rightarrow \mathbb{R}_{\geq 0}$ be some metric.

Then, the map matching error $e_m \in \mathbb{R}^2$ is defined by

$$e_m = \begin{cases} [m_d(\mathbf{d}_{\{t_1, t_n\}}, \mathbf{d}_{\mathcal{M}}), m_p(\mathbf{r}_{\{t_1, t_n\}}, \mathbf{r}_{\mathcal{M}})]^\top & \text{if } (\mathcal{F}_{C, \{t_1, t_n\}}, \mathcal{F}_{\mathcal{M}}) \text{ is a map match,} \\ \mathbf{c} & \text{otherwise,} \end{cases}$$

with some penalty vector $\mathbf{c}^\top \in \mathbb{R}_{>0}^2$.

Optionally, both components of the map matching distance can be combined into a one-dimensional map matching error by forming a weighted sum of them. A map matching error is only defined for either point OFAs and point MFAs or polygonal chain OFAs and polygonal chain MFAs.

The map matching error is computed for every shifted OFA to every MFA. To determine the best shift of all distances, the total map matching error is considered. Its definition is specified in the following.

Definition 5.10 (Total Map Matching Error). *Let there be k On-board Feature Area clusters determined with a descriptor function, let there be l Map Feature Areas, and let $e_{m,1,1}, \dots, e_{m,k,l} \in \mathbb{R}^2$ be their map matching errors. Then, the total map matching error $e_{tot\ m} \in \mathbb{R}^2$ is defined by*

$$e_{tot\ m} = \frac{1}{k} \sum_{i=1}^k \frac{1}{l} \sum_{j=1}^l e_{m,i,j}.$$

Optionally, both components of the total map matching error can be combined to a one-dimensional total map matching error by forming a weighted sum of them.

Then, the best shift of all Euclidean distances, here called residual, equals the Euclidean distance corresponding to the minimal total map match error over all shifts.

This is performed for every rotation and translation of certain intervals. The rotation and translation with the smallest residual represent the best rotation and translation between the OFA clusters and MFAs. These are carried out to find map matches.

Afterwards, the map matches are determined in the third step by applying Definition 5.8 to the rotated and translated OFA clusters and MFAs.

This rather expensive procedure is a major advantage as it enables the handling of small common intersection of both sets. This can be the case when OFA clusters are missing or are falsely detected. Missing OFA clusters often occur due to line-of-sight blockages, while falsely detected OFA clusters might arise, for example, when parking vehicles are detected as their descriptions resemble a wall.

The determination of map matches is performed for every time step independently. That means that the map matches do not depend on previous map matches. While repeating the map matching process once for every time step, the map matching process yields various sets of map matches over time. Not all elements of these map match sets are equal. Since these map matches may occur only once or a few times over time, they are not fed directly into the optimization problem. Also, OFA clusters that appear just once or a few times over time generate computational effort without gaining reliable localization information. Therefore, only valid map matches are applied in the optimization. Valid map matches are defined in the following.

Definition 5.11 (Valid Map Match). *Let $\mathcal{F}_{C, \{t_1, t_1\}}, \dots, \mathcal{F}_{C, \{t_1, t_k\}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ be the same On-board Feature Area cluster with either a point or polygonal chain position representation transformed into the UTM system at k different time steps $t_1, \dots, t_k \in \mathbb{R}$ determined with a descriptor function, let $\mathcal{F}_{\mathcal{M}} \in \mathbb{R}^{2^+} \times \mathbb{R}^m$ be a fixed Map Feature Area, let $(\mathcal{F}_{C, \{t_1, t_1\}}, \mathcal{F}_{\mathcal{M}}), \dots, (\mathcal{F}_{C, \{t_1, t_k\}}, \mathcal{F}_{\mathcal{M}})$ be l map matches determined in different time steps, and let $c_1 \in \mathbb{N}$ and $c_2 \in \mathbb{N}$ be some threshold. The map match $(\mathcal{F}_{C, \{t_1, t_i\}}, \mathcal{F}_{\mathcal{M}})$ is valid if it fulfills the following conditions*

- (i) $k > c_1$, (ii) $l > c_2$.

That means only those map matches are valid when often enough OFA clusters are matched with the same MFA and the OFA cluster contains enough associated OFAs. This is checked for every MFA individually. This validity check's advantage is that map matches not found in the current cycle can still be included in the optimization even if they were missed in the current time step.

Additionally, map matches can be revised through this process. If it was found later in time that an OFA cluster was matched more often to a different MFAs than in preceding time steps, the map matches are changed to the current more likely choice.

The set of all valid map matches $\mathcal{M} \subset \mathcal{P}(\mathcal{P}(\mathbb{R}^{2^+} \times \mathbb{R}^m) \times (\mathbb{R}^{2^+} \times \mathbb{R}^m))$ of each time step and the possible corrected map matches are fed into optimization, thus, solving the localization problem for every time step.

5.2.4. Graph Building

After generating the map with MFAs, extracting OFAs, associating the OFAs, and matching the associated OFAs with MFAs, the factor graph can be built. The factor graph is an intuitive way to illustrate the localization optimization problem. The goal of this phase is to represent all available information in the graph. This is done in every time step.

As described in Subsection 2.2.8, the graph consists of nodes and factors. All variables of the state vector are represented as nodes. That includes 2D vehicle poses. As the optimization should determine these, they are initialized in two different ways, depending on two cases. The initial pose is set with a GPS measurement. The subsequent poses are specified initially with the preceding pose, shifted, and rotated with the odometry data. Hereby, the odometry data constraints two consecutive poses as factors with its Euclidean distance and the angular difference between two vehicle poses. The GPS poses constrain the vehicle poses as unary factors and their Euclidean distances and the angular differences to them whenever available. The core information of the factor graph represent the valid map matches. Point OFA cluster representatives are added to the graph as nodes, constraining the vehicle poses by their distance in the Euclidean space to the vehicle poses without considering the vehicle's heading. The MFAs positions are included in the factor graph as unary factors limiting the point OFA cluster representatives with a distance in the Euclidean space combined with a descriptor specific distance in the description space. In the case of polygonal chain OFA cluster representatives, unary factors are implemented.

In this way, the factor graph is composed of the vehicle's odometry, GPS measurements, and valid map matches.

Figure 5.2 shows a sample factor graph representation applying the proposed method using the FPFH descriptor with a real-world data set in a suburban scenery, refer to Subsection 3.1.2 for the data set and to Subsection 5.1.2 for the applied sensors. MFAs are illustrated in violet with lines connecting multiple points, GPS factors as green triangles, the vehicle trajectory as orange triangles, the connections from the vehicle poses to MFAs corresponding to valid map matches as green lines, the estimated vehicle pose as orange rectangle, the GPS pose as green rectangle, and the pose of the reference system as blue rectangle, i.e., the vehicle pose measured with the Applanix POS LV 520 system. MFAs were extracted from dense point clouds as described in the previous chapter. The GPS data is recorded with a close-to-production NovAtel FlexPak-G2 OEMStar sensor. The odometry data from the EgoMaster modules was applied. OFAs were extracted from Velodyne VLP 32-C [150] point clouds using the FPFH descriptor. It can be seen that multiple valid map matches between observed OFAs and MFAs can be found. They constrain together with the odometry and GPS data the vehicle trajectory, whose current pose is close to that of the reference position.

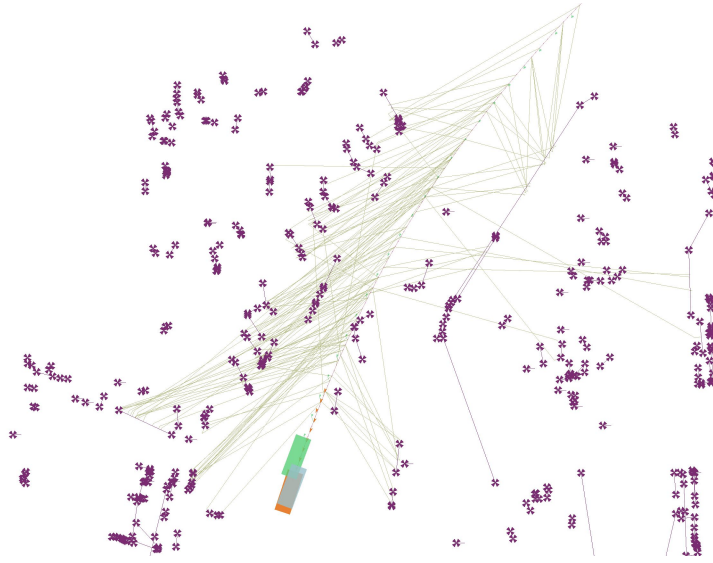


Figure 5.2: Representation of the localization optimization problem as a sample factor graph in a real-world data set on the example of the PPFH descriptor. The MFAs are represented with the symbol \ast (the points of polygonal chains are connected with a violet line), the GPS factors as \blacksquare , the vehicle trajectory as \blacktriangleleft , the connections from the vehicle poses to MFAs corresponding to valid map matches as green lines, the current vehicle pose as \blacksquare , the current GPS pose as \blacksquare , and the current pose of the reference system as \blacksquare .

5.2.5. Graph Optimization

In the optimization, the optimal state vector $\mathbf{x} = [\chi_{v,t_1}, \dots, \chi_{v,t_j}, \chi_{c,\{t_a,t_b\}}, \dots, \chi_{c,\{t_c,t_d\}}]$ should be found, i.e., j poses of the vehicle $\chi_{t_1}, \dots, \chi_{t_j} \in \mathbb{R}^2 \times [0, 2\pi)$ and d point OFAs cluster representatives $\mathbf{r}_{c,\{t_a,t_b\}}, \dots, \mathbf{r}_{c,\{t_c,t_d\}} \in \mathbb{R}^2$. That means, in this step, the optimization problem Equation 2.53 should be solved. This is done by finding the maximum a posteriori solution \mathbf{x}^* maximizing the probability $p(\mathbf{x}|\mathbf{z})$, where \mathbf{z} contains all measurements.

In Subsection 2.2.8, it has been shown that finding the maximum a posteriori solution can be simplified. For simplification, error terms are introduced. The error terms represent the deviation between the actual measurement \mathbf{z} and the predicted measurement $h(\mathbf{x})$ with h as a (nonlinear) function that predicts the measurement \mathbf{x} :

$$e(h(\mathbf{x}), \mathbf{z}) = h(\mathbf{x}) - \mathbf{z}. \quad (5.2)$$

These terms are defined for every data type applied in the optimization problem. In the following, e^g denotes the error terms between a GPS and a state vector's vehicle pose with GPS measurement uncertainties Ω^g given by the GPS sensor. e^o denotes the error terms between the relative poses of two vehicle poses of the state vector compared to odometry data with odometry measurement uncertainties Ω^o given by the EgoMaster. e^{fp} denotes the error terms between a state vector's points of OFA cluster representative and vehicle pose compared to the measured distance from the vehicle to the point OFA cluster representative with experimentally determined measurement uncertainties of the OFA detection Ω^{fp} . e^{fc} denotes the error terms between the distance of a state vector's vehicle pose to a polygonal chain MFA and a measured polygonal chain OFA cluster representative with experimentally determined measurement uncertainties of the OFA detection Ω^{fc} . Finally, e^m denotes the error terms between a matched MFA and the corresponding state vector's point OFA cluster representative with map generation uncertainties Ω^m .

With the error terms' introduction, finding the maximum a posteriori solution can be simplified to finding the state vector which minimizes the sum of the weighted squared error terms:

$$\begin{aligned}
\mathbf{x}^* &= \underset{\mathbf{x}}{\operatorname{argmax}} p(\mathbf{x}|\mathbf{z}) \\
&\approx \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i e^o(\mathbf{x}, \mathbf{z}_i^o)^T \Omega_i^o e^o(\mathbf{x}, \mathbf{z}_i^o) \\
&\quad + \sum_i e^g(\mathbf{x}, \mathbf{z}_i^g)^T \Omega_i^g e^g(\mathbf{x}, \mathbf{z}_i^g) \\
&\quad + \sum_i e^{fp}(\mathbf{x}, \mathbf{z}_i^{fp})^T \Omega_i^{fp} e^{fp}(\mathbf{x}, \mathbf{z}_i^{fp}) \\
&\quad + \sum_i e^{fc}(\mathbf{x}, \mathbf{z}_i^{fc})^T \Omega_i^{fc} e^{fc}(\mathbf{x}, \mathbf{z}_i^{fc}) \\
&\quad + \sum_i e^m(\mathbf{x}, \mathbf{z}_i^m)^T \Omega_i^m e^m(\mathbf{x}, \mathbf{z}_i^m),
\end{aligned} \tag{5.3}$$

where the individual error terms are weighted with the information matrix Ω_i of the i -th measurement as the inverse of the measurement's covariance matrix C_i , i.e., measurement uncertainties.

The practical solution of Equation 5.3 is gained through the numerical iterative Gauss-Newton method, refer to Subsection 2.2.8. For the numerical solution, the error terms are explicitly defined in the following together with their Jacobian matrices, cf. Equation 2.55, for each data type. As a reminder, the Jacobian matrices consist of the partial derivatives of the error function according to the state vector.

Between Point MFA and Point OFA cluster representative: Let $\mathcal{F}_M = (\mathbf{r}_M, \mathbf{d}_M) \in \mathbb{R}^2 \times \mathbb{R}^m$ be a point MFA specified in the UTM system, let $\mathcal{F}_{C, \{t_1, t_k\}} \subset \mathbb{R}^2 \times \mathbb{R}^m$ be a measured On-board Feature Area cluster with a point cluster representative with k different time steps $t_1, \dots, t_k \in \mathbb{R}$, let $(\mathcal{F}_{C, \{t_1, t_k\}}, \mathcal{F}_M)$ be a valid map match, and let $\mathcal{F}'_{C, \{t_1, t_k\}} \subset \mathbb{R}^2 \times \mathbb{R}^m$ be an On-board Feature Area cluster with a point cluster representative $\mathcal{C}'_{O, \{t_1, t_k\}} = (\chi_{O, \{t_1, t_k\}}, \mathbf{d}'_{O, \{t_1, t_k\}})$ of the state vector specified in the UTM system with k different time steps $t_1, \dots, t_k \in \mathbb{R}$.

Then, the error terms $e^m(\mathbf{r}_M, \chi_{O, \{t_1, t_k\}}) \in \mathbb{R}^2$ between the measured point MFA and the point OFA cluster representative of the state vector are defined as:

$$e^m(\mathbf{r}_M, \chi_{O, \{t_1, t_k\}}) = \chi_{O, \{t_1, t_k\}} - \mathbf{r}_M. \tag{5.4}$$

The error terms are defined as the difference between the global point OFA cluster representative and the point MFA. That means that the point OFA is constrained by the point MFA cluster representative, i.e., when optimizing the graph, both positions should be as similar as possible.

The resulting Jacobian matrix $\mathbf{J}^m \in \mathbb{R}^{2 \times \dim(\mathbf{x})}$ consists of the identity matrix $\mathbb{1}^{2 \times 2} \in \mathbb{R}^{2 \times 2}$ at the columns belonging to $\chi_{O, \{t_1, t_k\}}$. The columns with the partial derivatives according to the other elements of the state vector contain only zero entries:

$$\mathbf{J}^m = [\mathbf{0} \quad \dots \quad \mathbb{1}^{2 \times 2} \quad \dots \quad \mathbf{0}]. \tag{5.5}$$

This is due to the fact that the point OFA cluster representative is part of the state vector.

Between Absolute Poses and Vehicle Poses: Let $\mathbf{z}_g = [x_g, y_g, \theta_g] \in \mathbb{R}^2 \times [0, 2\pi)$ be an absolute pose measured in the UTM coordinate system, e.g., a GPS pose, and let $\chi_v = (x_v, y_v, \theta_v) \in \mathbb{R}^2 \times [0, 2\pi)$ be the vehicle pose of the state vector specified in the UTM coordinate system. The measurement uncertainties of the absolute pose is specified in the VRF while the measurement of the absolute pose is defined in the UTM coordinate system. The angle error does not depend on the reference system

and does not need to be transformed. For this, the error terms between the GPS and the vehicle pose are specified in the VRF by rotating the poses difference with the rotation matrix $\mathbf{R}_g^T \in \mathbb{R}^{3 \times 3}$ with a measured vehicle heading θ_g . Then, the error terms $e^g(\chi_v, z_g) \in \mathbb{R}^3$ between a GPS and a vehicle pose are defined as:

$$e^g(\chi_v, z_g) = \mathbf{R}_g^T(\chi_v - z_g), \text{ with } \mathbf{R}_g^T = \begin{bmatrix} \cos(\theta_g) & \sin(\theta_g) & 0 \\ -\sin(\theta_g) & \cos(\theta_g) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.6)$$

Here, the error terms are defined as the difference between the absolute and the vehicle pose, which is rotated into the VRF. That means that the vehicle pose is constrained by the measured absolute pose, i.e., when optimizing the graph, both poses should be as similar as possible. However, here, the transformation of the UTM coordinate system into the VRF needs to be taken into account.

Consequently, the Jacobian matrix $\mathbf{J}^g \in \mathbb{R}^{3 \times \dim(x)}$ consists of the rotation matrix \mathbf{R}_g^T at the columns belonging to χ_v :

$$\mathbf{J}^g = [\mathbf{0} \quad \dots \quad \mathbf{R}_g^T \quad \dots \quad \mathbf{0}]. \quad (5.7)$$

Between Relative Vehicle Poses Compared to Odometry: Let $\chi_{v,t_i} = [t_{v,t_i}, \theta_{v,t_i}]$, $\chi_{v,t_{i+1}} = [t_{v,t_{i+1}}, \theta_{v,t_{i+1}}] \in \mathbb{R}^2 \times [0, 2\pi)$ be vehicle poses of two consecutive time steps $t_i, t_{i+1} \in \mathbb{R}$ of the state vector in the UTM coordinate system and let $z_{o,\{t_i,t_{i+1}\}} = [t_{o,\{t_i,t_{i+1}\}}, \theta_{o,\{t_i,t_{i+1}\}}] \in \mathbb{R}^2 \times [0, 2\pi)$ be the odometry measurement between the two consecutive vehicle poses measured in the VRF of χ_{v,t_i} . Here, the idea of the error terms is to compare the odometry measurement with the relative vehicle pose of the state vector's two consecutive poses.

Therefore, the relative vehicle pose $\chi_{v,\{t_i,t_{i+1}\}} = [t_{v,\{t_i,t_{i+1}\}}, \theta_{v,\{t_i,t_{i+1}\}}] \in \mathbb{R}^2 \times [0, 2\pi)$ between the consecutive poses is calculated as the difference between them:

$$\chi_{v,\{t_i,t_{i+1}\}} = \begin{bmatrix} t_{v,t_{i+1}} \\ \theta_{v,t_{i+1}} \end{bmatrix} - \begin{bmatrix} t_{v,t_i} \\ \theta_{v,t_i} \end{bmatrix}. \quad (5.8)$$

As both vehicle poses are defined in the global UTM coordinate system but the odometry is measured in the VRF, the relative vehicle pose is rotated into the VRF by rotating it with the transposed rotation matrix $\mathbf{R}_{v,t_i}^T \in \mathbb{R}^{3 \times 3}$:

$$\chi'_{v,\{t_i,t_{i+1}\}} = \mathbf{R}_{v,t_i}^T \chi_{v,\{t_i,t_{i+1}\}}, \text{ with } \mathbf{R}_{v,t_i} = \begin{bmatrix} \cos(\theta_{t_i}) & -\sin(\theta_{t_i}) & 0 \\ \sin(\theta_{t_i}) & \cos(\theta_{t_i}) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.9)$$

Then, the difference between the relative vehicle poses and the odometry measurement leads to the error terms $e^o(\chi_{v,t_i}, \chi_{v,t_{i+1}}, z_{o,\{t_i,t_{i+1}\}})$. Since the measurement uncertainties of the odometry are defined in the VRF, the difference is rotated with the rotation matrix $\mathbf{R}_o^T \in \mathbb{R}^{2 \times 2}$:

$$\begin{aligned} e^o(\chi_{v,t_i}, \chi_{v,t_{i+1}}, z_{o,\{t_i,t_{i+1}\}}) &= \mathbf{R}_o^T(\chi'_{v,\{t_i,t_{i+1}\}} - z_{o,\{t_i,t_{i+1}\}}) \\ &= \begin{bmatrix} \mathbf{R}_o^T(\mathbf{R}_{v,t_i}^T(t_{v,t_{i+1}} - t_{v,t_i}) - t_{o,\{t_i,t_{i+1}\}}) \\ \theta_{v,t_{i+1}} - \theta_{v,t_i} - \theta_{o,\{t_i,t_{i+1}\}} \end{bmatrix}, \\ &\text{with } \mathbf{R}_o = \begin{bmatrix} \cos(\theta_{o,\{t_i,t_{i+1}\}}) & -\sin(\theta_{o,\{t_i,t_{i+1}\}}) \\ \sin(\theta_{o,\{t_i,t_{i+1}\}}) & \cos(\theta_{o,\{t_i,t_{i+1}\}}) \end{bmatrix}. \end{aligned} \quad (5.10)$$

That means that two consecutive vehicle poses are constrained by the odometry measurement, i.e., when optimizing the graph, both difference between the vehicle poses should be as similar as pos-

sible to the odometry. However, here, the transformation of the vehicle poses defined in the UTM coordinate system into the VRF of the first vehicle pose needs to be taken into account.

The dependence on multiple poses also has an effect on the Jacobian matrix $\mathbf{J}^o \in \mathbb{R}^{3 \times \dim(\mathbf{x})}$. The partial derivatives according to the two poses are represented by the respective partial matrices. Except for them, the Jacobian matrix consists only of zero entries:

$$\mathbf{J}^o = \begin{bmatrix} \mathbf{0} & \dots & \frac{\partial \mathbf{e}^o}{\partial \mathbf{x}_{v,t_i}} & \frac{\partial \mathbf{e}^o}{\partial \mathbf{x}_{v,t_{i+1}}} & \dots & \mathbf{0} \end{bmatrix}, \quad (5.11)$$

$$\text{with } \frac{\partial \mathbf{e}^o}{\partial \mathbf{x}_{v,t_i}} = \begin{bmatrix} \mathbf{R}_o^\top \mathbf{R}_{v,t_i}^\top & \mathbf{R}_o^\top \frac{\partial \mathbf{R}_{v,t_i}^\top}{\partial \theta_{v,t_i}} (\mathbf{t}_{v,t_{i+1}} - \mathbf{t}_{v,t_i}) \\ \mathbf{0} & -1 \end{bmatrix} \text{ and } \frac{\partial \mathbf{e}^o}{\partial \mathbf{x}_{v,t_{i+1}}} = \begin{bmatrix} \mathbf{R}_o^\top \mathbf{R}_{v,t_i}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}.$$

Between Vehicle Pose and Point OFA Cluster Representative: Let $\mathbf{x}_{v,t_i} = [t_{v,t_i}, \theta_{v,t_i}] \in \mathbb{R}^2 \times [0, 2\pi)$ be a vehicle pose of the state vector, let $\mathcal{F}_{C,\{t_1,t_k\}} \subset \mathbb{R}^2 \times \mathbb{R}^m$ be an On-board Feature Area cluster with a point cluster representative $(\mathcal{X}_{C,\{t_1,t_k\}}, \mathbf{d}_{C,\{t_1,t_k\}})$ of the state vector specified in the UTM system with k different time steps $t_1, \dots, t_k \in \mathbb{R}$, let $\mathcal{F}'_{C,\{t_1,t_k\}} \subset \mathbb{R}^2 \times \mathbb{R}^m$ be a measured On-board Feature Area cluster with a point cluster representative $(\mathbf{r}'_{C,\{t_1,t_k\}}, \mathbf{d}'_{C,\{t_1,t_k\}})$ specified in the VRF with k different time steps $t_1, \dots, t_k \in \mathbb{R}$. In this case, the main idea of the error terms is to compare the position of the measured OFA cluster representative, specified in the VRF, with the position difference between the state vector's OFA cluster representative and its vehicle pose, both specified in the UTM system.

Therefore, the difference χ_{diff} between the positions of the state vector's OFA cluster representative $\mathcal{X}_{C,\{t_1,t_k\}}$ and its vehicle position \mathbf{t}_{v,t_i} is determined and rotated into the VRF:

$$\chi_{\text{diff}} = \mathbf{R}_{v,t_i}^\top (\mathcal{X}_{C,\{t_1,t_k\}} - \mathbf{t}_{v,t_i}), \text{ with } \mathbf{R}_{v,t_i} = \begin{bmatrix} \cos(\theta_{v,t_i}) & -\sin(\theta_{v,t_i}) \\ \sin(\theta_{v,t_i}) & \cos(\theta_{v,t_i}) \end{bmatrix}. \quad (5.12)$$

Since the measurement uncertainties are also specified in the VRF, a transformation does not need to be performed. Then, the difference between the rotated position difference and the measured OFA cluster representative yields the error terms $e^{fp}(\mathbf{x}_{v,t_i}, \mathcal{X}_{C,\{t_1,t_k\}}, \mathbf{r}'_{C,\{t_1,t_k\}})$:

$$e^{fp}(\mathbf{x}_{v,t_i}, \mathcal{X}_{C,\{t_1,t_k\}}, \mathbf{r}'_{C,\{t_1,t_k\}}) = \chi_{\text{diff}} - \mathbf{r}'_{C,\{t_1,t_k\}} = \mathbf{R}_{v,t_i}^\top (\mathcal{X}_{C,\{t_1,t_k\}} - \mathbf{t}_{v,t_i}) - \mathbf{r}'_{C,\{t_1,t_k\}}. \quad (5.13)$$

The idea of this error term is comparable to that of the odometry measurements. That means that the difference between the state vector's vehicle pose and a point OFA cluster representative is constrained by the measured point OFA cluster representative, i.e., when optimizing the graph, both the difference and the measured point OFA cluster representative should be as similar as possible.

Since the error terms depend on the vehicle pose and the OFA cluster representative of the state vector (not the measured one), there are only entries in the corresponding columns of the Jacobian matrix $\mathbf{J}^{fp} \in \mathbb{R}^{2 \times \dim(\mathbf{x})}$:

$$\mathbf{J}^{fp} = \begin{bmatrix} \mathbf{0} & \dots & \frac{\partial e^{fp}}{\partial \mathbf{x}_{v,t_i}} & \dots & \mathbf{0} & \dots & \frac{\partial e^{fp}}{\partial \mathcal{X}_{C,\{t_1,t_k\}}} & \dots & \mathbf{0} \end{bmatrix}, \quad (5.14)$$

$$\text{with } \frac{\partial e^{fp}}{\partial \mathbf{x}_{v,t_i}} = \begin{bmatrix} -\mathbf{R}_{v,t_i}^\top & \frac{\partial \mathbf{R}_{v,t_i}^\top}{\partial \theta_{v,t_i}} (\mathcal{X}_{C,\{t_1,t_k\}} - \mathbf{t}_{v,t_i}) \end{bmatrix} \in \mathbb{R}^{2 \times 3} \text{ and } \frac{\partial e^{fp}}{\partial \mathcal{X}_{C,\{t_1,t_k\}}} = [\mathbf{R}_{v,t_i}^\top] \in \mathbb{R}^{2 \times 2}.$$

Between Vehicle Pose and Point of Polygonal Chain MFA: Let $\chi_{v,t_i} = [\mathbf{t}_{v,t_i}, \theta_{v,t_i}] \in \mathbb{R}^2 \times [0, 2\pi)$ be a vehicle pose of the state vector, let $\mathcal{F}'_{\mathcal{C},\{t_1,t_k\}} \subset \mathbb{R}^{2^+} \times \mathbb{R}^m$ be a measured On-board Feature Area cluster with a polygonal chain cluster representative $(\mathbf{r}'_{\mathcal{C},\{t_1,t_k\}}, \mathbf{d}'_{\mathcal{C},\{t_1,t_k\}})$ specified in the VRF with j points $\mathbf{r}'_{\mathcal{C},\{t_1,t_k\}} = \{\mathbf{r}'_{\mathcal{C},\{t_1,t_k\},1}, \dots, \mathbf{r}'_{\mathcal{C},\{t_1,t_k\},j}\}$ k different time steps $t_1, \dots, t_k \in \mathbb{R}$, and let $\mathcal{F}_{\mathcal{M}} \in \mathbb{R}^2 \times \mathbb{R}^m$ be a polygonal chain MFA specified in the UTM system, and let $(\mathcal{F}'_{\mathcal{C},\{t_1,t_k\}}, \mathcal{F}_{\mathcal{M}})$ be a valid map match.

Let $\mathbf{v}_m = [v_{m,x}, v_{m,y}] \in \mathbb{R}^2$ be the unit vector used for the orientation of n line segments of the MFA. In the error terms' determination, each point of the polygonal chain OFA cluster representative and each line segment of the polygonal chain MFA is considered individually. Therefore, for each point and line segment of the polygonal chain, the error terms $e^{fc}(\chi_{v,t_i}, \mathbf{r}'_{\mathcal{C},\{t_1,t_k\},l}, \mathcal{F}_{\mathcal{M}})$ are determined by rotating the OFA cluster representative into the UTM system and building the orthogonal distance between the OFA cluster representative and the MFA. The error terms determine the minimum distance between the measured point and the corresponding line segment:

$$e^{fc}(\chi_{v,t_i}, \mathbf{r}'_{\mathcal{C},\{t_1,t_k\},l}, \mathcal{F}_{\mathcal{M}}) = (\mathbf{R}_{v,t_i} \mathbf{r}'_{\mathcal{C},\{t_1,t_k\},l} + \mathbf{t}_{v,t_i})^\top \cdot \begin{bmatrix} v_{m,y} \\ -v_{m,x} \end{bmatrix}, \quad (5.15)$$

$$\text{with } \mathbf{R}_{v,t_i} = \begin{bmatrix} \cos(\theta_{t_i}) & -\sin(\theta_{t_i}) \\ \sin(\theta_{t_i}) & \cos(\theta_{t_i}) \end{bmatrix}.$$

Similar to the error term before, the idea here is that the difference between the state vector's vehicle pose and a point of a polygonal chain OFA cluster representative is constrained by the measured point of a polygonal chain OFA cluster representative, i.e., when optimizing the graph, both the difference and the measured point of a polygonal chain OFA cluster representative should be as similar as possible.

Since the error terms are only dependent on the vehicle pose the Jacobian matrix $\mathbf{J}^{fc} \in \mathbb{R}^{1 \times \dim(\mathbf{x})}$ becomes:

$$\mathbf{J}^{fc} = \begin{bmatrix} \mathbf{0} & \dots & v_{m,y} & -v_{m,x} & \left(\frac{\partial e^{fc}}{\partial \theta_{t_i}}\right) & \dots & \mathbf{0} \end{bmatrix}, \quad (5.16)$$

$$\text{with } \frac{\partial e^{fc}}{\partial \theta_{t_i}} = \left(\begin{bmatrix} \cos(\theta_{t_i}) & -\sin(\theta_{t_i}) \\ \sin(\theta_{t_i}) & \cos(\theta_{t_i}) \end{bmatrix} \cdot \mathbf{r}'_{\mathcal{C},\{t_1,t_k\},l} \right)^\top \cdot \begin{bmatrix} v_{m,y} \\ -v_{m,x} \end{bmatrix}.$$

5.2.6. Sliding Window

The computational effort when solving the optimization problem depends on the number of elements of the state vector. The state vector is the vector containing all OFAs and vehicle poses. Thus, the computational effort largely depends on the number of OFAs and poses. The more OFA variables, which are available to minimize the weighted error squares, the more complex the problem is.

In the real-time application, the variables of the optimization problem are supplemented by new vehicle poses with the associated measurements in each time step. If measurements of OFA are detected for the first time in a time step, they can be added to the optimization problem. As a result, the number of elements in the state vector grows. In order to enable a real-time capability, the complexity and thus the computing effort must be limited.

Therefore, the sliding window approach is introduced to limit the number of OFAs and poses. As the OFA measurements are associated with the poses, the number of elements in the state vector is controlled with the number of poses. If the limit value is reached, the oldest pose with the associated OFA measurements is removed for each time step. The same applies to OFAs which are no longer linked to any poses by measurements.

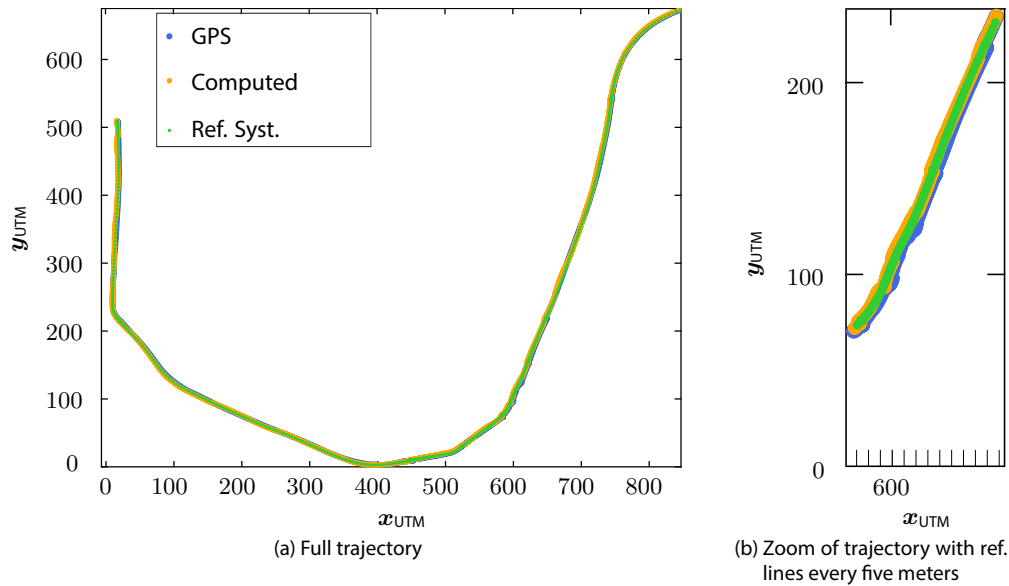


Figure 5.3: Vehicle trajectory computed with the method proposed in this thesis compared to a reference trajectory and a GPS trajectory on the example of the FPFH descriptor

5.3. Analyzes

In the following, the non-semantic localization matching OFAs with MFAs is analyzed. Therefore, the position accuracy of this approach is evaluated. Also an outlook on the computation effort computing this localization is given.

5.3.1. Position Accuracy

The experimental evaluation is designed to evaluate the proposed graph-based localization using non-semantic features on real-world data sets by determining position deviations compared to reference positions. It investigates Requirement 5.3 (i).

The first step of localization, *Feature Area Collection*, is realized in the following way. First, the map is created by extracting MFAs from the dense and globally referenced point cloud data set generated by accumulating multiple records of a Trimble MX8 LiDAR sensor. Second, the OFAs are collected from the sparse on-board point clouds recorded with a Velodyne VLP 32-C LiDAR sensor. Both extractions are performed as described in Chapter 4. Here, 60% of the FAs according to the LIG value are extracted as OFAs and MFAs. In this experiment, three geometry-based descriptors are applied, the FPFH, the SHOT, and the RoPS. Therefore, the descriptor-specific distinctiveness measures are applied according to Equations 4.8, 4.10, and 4.12.

In the second step of localization, *Association of Feature Areas*, on-board detected data is clustered according to Subsection 5.2.2. Additionally, some metrics and thresholds need to be defined to associate OFAs. For comparing positions of polygonal chains containing at least two points, the positions of orthogonal projections of the points of the polygonal chains onto one another are compared. The metric is explained in detail in Subsection 5.2.2. If the polygonal chain consists only of one point,

Table 5.2: Absolute position errors of the LiDAR-feature-based localization method averaged over a test drive in an urban scenery of 1.6 kilometers. The ground truth is an Applanix POS LV 520 system. The errors are compared to the average absolute position errors measured with a close-to-production GNSS OEMStar.

	absolute [m]	lateral [m]	longitudinal [m]
FPFH	0.45 ± 0.37	0.23 ± 0.24	0.41 ± 0.28
SHOT	0.51 ± 0.41	0.24 ± 0.28	0.47 ± 0.42
RoPS	0.46 ± 0.43	0.22 ± 0.3	0.4 ± 0.32
OEMStar	1.1 ± 0.74	0.73 ± 0.59	0.93 ± 0.46

the Euclidean distance is applied. For comparing the descriptions, descriptor-specific (normalized) Euclidean distances are used according to Equations 3.2, 3.3, and 3.4. The maximum Euclidean distance for a point OFA cluster and a point OFA to be associated is set to two meters, for polygonal chain OFA clusters to three and a half meters. These thresholds were determined to be suitable in real-world data analyzing multiple test drives, refer to Subsection 3.1.2 for the description of the used data set. The maximum (normalized) Euclidean distance for the FPFH descriptor is set to 0.2, for the SHOT to 0.25, and for the values of the RoPS descriptor to 0.01, 0.2, 0.2, 0.15, 0.1. These thresholds were set based on practical testing with the data set according to Table 3.2.

In the third step of localization, *Map Matching*, the clustered OFAs are matched to MFAs according to Subsection 5.2.3. Equally to the preceding step, some metrics and thresholds are specified, which are used in this experiment. For all metrics, the same ones are applied in the map matching step as in the association of FAs step. The same description thresholds are used for each descriptor. However, different thresholds are chosen for the metrics comparing the spatial information. For the point representations, a maximal distance of one meter is set. For the polygonal chain representations, a maximal distance of 0.75 meters is set. A map match is considered valid if the map match consists of an OFA cluster containing at least three OFAs and a MFA, with which this OFA cluster was matched at least three times, refer to Definition 5.11. The graph build with this data as a sliding window length of 100 vehicle poses and OFA. These values were identified based on practical experiments with the data set according to Table 3.2.

The fourth step of localization, *Optimization*, is performed using *g2o*, which is an open-source C++ framework [67]. *g2o* is designed for optimizing graph-based nonlinear error functions, where in this thesis the Gauss-Newton method of *g2o* was employed. This outputs the actual vehicle pose applying all steps of the *LiDAR-Feature-based Localization*.

The test data was collected driving the trajectory in an urban scenery. The odometry data of the test vehicle is measured with the software module *EgoMaster*. Refer to Subsection 5.1.2 for the sensors of the test vehicle seen in Figure 3.1. The test drive has a total length of approximately 1.6 kilometers and includes velocities from minimal $0 \frac{m}{s}$ to maximal $16.6 \frac{m}{s}$. See Figure 5.3 for a visualization of the trajectory. Within this experiment, the accuracy of the calculated poses is measured with an Applanix POS LV 520 system serving as a reference. Additionally, as a comparison, the accuracy of a close-to-production GPS, i.e., NovAtel FlexPak-G2 OEMStar sensor, is shown.

The results of this experiment are shown in 5.2. It demonstrates that the approach of this thesis yields RMS position errors which are in a range of 0.4 to 0.5 meters, which outperforms the position calculation of the OEMStar accuracy of 1.1 meters. The mean absolute lateral errors are smaller compared to the longitudinal errors. This is due to the fact that in the test drive more polygonal chain FAs could be matched compared to point FAs. Elongated polygonal chain FAs provide more information about the lateral vehicle position compared to the longitudinal position.

Compared to the requirements of Reid et al. [104], i.e., that in 95% of the cases 0.1 meters accuracy is reached, it can be seen that localization implemented in this thesis is not accurate enough. There are several reasons for this. Since the map, i.e., the MFAs, is generated automatically based on raw dense point clouds, which still include dynamic objects like parking vehicles or pedestrians, the map matching process includes systematic errors. Thus, the possible MFA with which the OFAs clusters can be matched are more complex and include non-persistent objects. For example, walls detected on-board might be matched with a parking truck of the map. Therefore, some map elements should be deleted from the map, e.g., using semantics like excluding vehicles from the map with an algorithm or manually by hand. Additionally, it can be discovered that the generated MFAs, i.e., the map, includes artifacts created during the process of the non-semantic map generation. These artifacts should be discarded in a next step, e.g., unifying two MFAs or discarding MFAs which look like clutter. Further, the C++ code implemented for this evaluation is in the state of a proof-of-concept code. For instance, it can be seen that the polygonal chain map matches are added to the factor graph very late. That means that it occurs sometimes that no map matches are in the graph if additionally no point FAs could be matched. In these cases, localization relies only on GPS and odometry information. This results in an estimated vehicle position which drifts off due to odometry inaccuracies. The drifting effect can be seen in Figure 5.3b. After a period where no map matches were found and the estimated vehicle position drifted, and then map matches are found, the estimated vehicle position jumps close to the reference position.

The experiment shows that the concept of a localization relying on non-semantic elements of the environment is practically relevant. In the next step, the implementation must be further refined and developed for practical use.

5.3.2. Demand for Computing Resources

In this experiment, the demand of computing resources of the *LiDAR-Feature-based Localization* are examined. However, this thesis focuses on the localization's potential, not on the real-time capability. This thesis pays attention to whether it is possible to compute an accurate vehicle pose based on non-semantic data, not on implementing it regarding computation time. Therefore, overall, the computation time plays a minor role in this thesis.

Two analyzes are carried out in this context. First, theoretical considerations on the dependencies of algorithms' complexity on the the input parameters are presented. After that, an empirical analysis measures the computation time of the implemented code of this thesis based on real-world data.

Theoretical Considerations:

The following theoretical evaluations consider the complexity in dependence of the input data. These estimations are carried out for each of the three work steps of this thesis, i.e., description calculation, selection, and localization.

At the beginning, the effort of the description calculation in dependence on the input data is estimated. The number of points within the local neighborhood of a query point as input for description calculation and the number of descriptions to be calculated within a point cloud are used as parameters for estimating the complexity. To do this, these sensor type specific indicators are compared to one another to estimate the dependency of the description calculation's complexity on the sensor. First, a synthetic scene is created and sampled with three different types of sensors. Second, real-world data sampled with two sensors with different sampling rates, resolutions and ranges is generated. Depending on the sensor characteristics, it is examined how the parameters and thus the complexity change.

For this purpose, a synthetic scene, as seen in Figure 3.2a, is sampled using a Velodyne VLP 16 [151], a Velodyne VLP 32-C [150], and a modified Velodyne VLP 16 model. The modified Velodyne VLP 16 model is based on the characteristics of the Velodyne VLP 16 but with 1° vertical resolution instead of 2° (thus,

Table 5.3: Average total number of points within the point clouds, average number of points within the preprocessed point clouds, and average number of points within the local neighborhoods for description calculation. The numbers were computed based on synthetic data (^s) with Velodyne VLP 16, Velodyne VLP 32-C, and a modified Velodyne VLP 16 models and based on real-world point clouds of an urban scenery (^r) recorded with the Velodyne VLP 16, the Velodyne VLP 32-C.

	total	preprocessed point cloud	local neighborhood
Velodyne VLP 16 ^s	23,470	17,074	349
Velodyne VLP 32-C ^s	48,293	36,506	107
mod. Velodyne VLP 16 ^s	45,518	38,804	620
Velodyne VLP 16 ^r	13,803	3,667	66
Velodyne VLP 32-C ^r	43,430	10,843	38

a higher resolution with 32 layers in the same vertical field of view). Then, the total number of points within the point clouds, the number of points within the preprocessed point clouds (points which provide suitable information for the description calculation, cf. Subsection 4.2.1), and the average number of points within the local neighborhoods around the query points of the preprocessed point cloud are determined dependent on the sensor. The same experiment is performed with real-world data. Here, real-world point clouds were recorded with the Velodyne VLP 16 and the Velodyne VLP 32-C along the same urban trajectory with 2760 point clouds. Then, the same indicators as before are calculated, however, the total number of points and the number of points of the preprocessed point clouds are averaged.

Table 5.3 presents the determined (average) number of points. Comparing the numbers computed with the synthetic data of the Velodyne VLP 16 and the modified Velodyne VLP 16 with double the resolution but same other sensor characteristics, it can be seen that with a higher sensor resolution all numbers become larger. The number of points in the local neighborhood, which indicates the complexity of the description calculation for one point, is nearly doubled with twice the resolution. The number of descriptions which need to be calculated for the synthetic scene is defined by the number of points of the preprocessed point cloud. Here, the same effect as for the number of points in the local neighborhood can be observed. Twice the resolution results in nearly twice the number of points in the preprocessed point cloud. Concluding, with double the resolution, twice as many descriptions have to be calculated and twice as many neighborhood points have to be considered per description. This leads to a quadratic influence of the resolution on the complexity.

However, another effect can be seen when comparing the results of the Velodyne VLP 16 to the Velodyne VLP 32-C for both, the synthetic and the real-world data. While the total number of points and number of points within the preprocessed point clouds of the Velodyne VLP 16 point cloud is significantly smaller compared to the Velodyne VLP 32-C point cloud, the number of points within the local neighborhoods of the preprocessed point clouds are approximately twice as large. This is due to the different vertical resolutions, opening angles and ranges of the sensors. The Velodyne VLP 16 has a vertical field of view of 30° and an equidistant vertical resolution. The Velodyne VLP 32-C has a vertical field of view of 40° and a non-equidistant vertical resolution with middle layers with a very high resolution but with outer layers which are far apart. The points on the outer layers of the Velodyne VLP 32-C are much further apart than the points on its inner layers and than the outer layers of the Velodyne VLP 16. This in total leads to a smaller number of points within the local neighborhoods. In practice, this effect is increased as the Velodyne VLP 32-C has a higher range (up to 200 meters) than the Velodyne VLP 16 (up to 100 meters). It is able to sparsely sample objects which are far away, which the Velodyne VLP 16 is not capable of. Hence, the Velodyne VLP 32-C preprocessed point cloud contains more points which are far away and thus neighborhoods which are sparsely sampled.

Regarding the complexity of the description calculation, it can be said that the larger the actual resolution, the larger the input for the description calculation and the more descriptions need to be calculated per point cloud. That means that a linear relation between the resolution and the input data for the calculation of one description can be assumed, but the complexity has a quadratic dependence of the sensor resolution for the total description calculation of the whole point cloud. For descriptors with linear complexity, e.g., the FPFH descriptor [110], this also means a linear increase in complexity for the actual description calculation for one point. For descriptors with quadratic complexity, e.g., the PFH descriptor [108–110], this means a quadratic increase in complexity for the actual description calculation for one point. Thus, the sensor resolution has severe influences on the computation time, even larger for descriptors with quadratic complexities like the PFH. However, it can be seen that not only the resolution but also the range and the distribution of the sensors' layers must be taken into account.

The complexity of the selection step is more difficult to estimate theoretically. As clustering algorithms are applied in the selection, the complexity largely depends on the algorithms' parameters. In the first part of the selection step, all calculated descriptions are grouped into two clusters in the description space applying the k -medoid, cf. Subsection 2.2.7. The effort involved depends heavily on the number of input descriptions and the length of the individual descriptors. As can be seen in the paragraph before, the number of input descriptions depends on the sensor resolution. The larger the resolution of a sensor, the larger the number of points of the preprocessed point cloud, i.e., the number of input descriptions, for the k -medoid clustering is. The size of the descriptions depends on the applied descriptor. For example, the FPFH descriptor [110] is 33-dimensional, the SHOT descriptor [112, 134, 136] is 352-dimensional, and the RoPS descriptor [43, 45] is 135-dimensional, as defined in the PCL [107]. The larger the size of the description, the more entries have to be compared with one another in the distance metric of the clustering. Additionally, the parameters of the k -medoid clustering need to be considered when discussing complexity. A maximum number of k -medoid clustering iterations of 100 has proven to be practicable for the application of this thesis, which represents a trade off between accuracy and effort. In the second part of the selection, the two description clusters are separated into many spatial clusters applying DBSCAN. That depends heavily on the number of points in the respective description cluster. Again, the number of points of the preprocessed point cloud is the decisive factor, which depends on the sensor resolution, as seen before. Additionally, the parameters used applying DBSCAN have an effect on the complexity. Here, a good trade off of an ε -environment of $\varepsilon = 0.5$ and a minimum number of points within the ε -environment of $m = 15$ has been discovered in empirical analyzes, cf. Subsection 2.2.6. The last part, the selection and representation of FAs, largely depends on the number of points of in each FAs and the number of selected FAs. There is a quadratic influence here. For each FAs, the 3D points must be represented as 2D points and polygonal chains. This means that the more FAs are extracted, the more points have to be put into the representation algorithm. As the number of FAs depends on the scenery, no assumptions are made about that here. But here, too, the sensor resolution is crucial when calculating the 2D representations, the effect of which was discussed in detail before.

For the estimation of the complexity of the localization, the number of elements in the factor graph used for localization are a good indicator. The optimization of the factor graph yields the estimated vehicle pose. Therefore, the number of nodes and factors in factor graphs was measured in a real test drive along a 1.6 kilometer long trajectory in an urban setting, cf. Figure 5.3. The required calculation time for localization, i.e., FA association, map matching and graph optimization, per vehicle pose estimation was set against this.

Figure 5.4 visualizes the results of this experiment as a box plot. The red line in each box presents the median, the lower edges of the boxes the 25%-percentile, the upper edges of the boxes the 75%-percentile, the whiskers the extreme data points (without considering outliers), and the red cross symbols the outliers. The number of nodes and factors of the graph are summarized in intervals of lengths

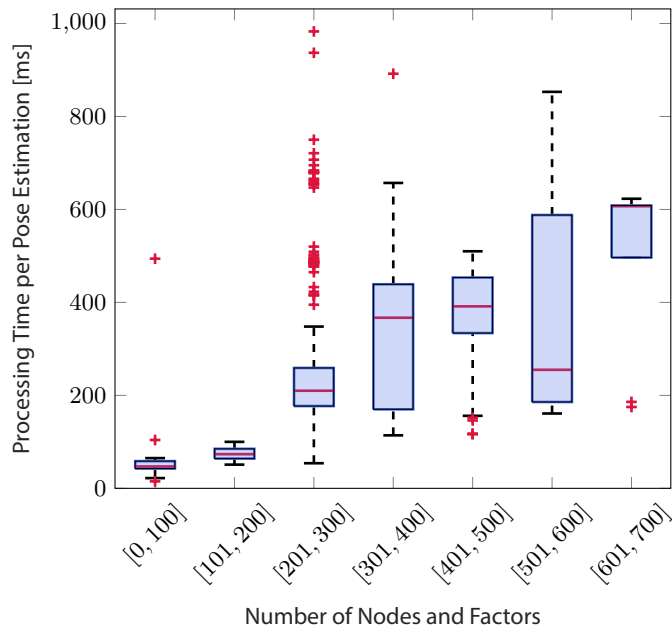


Figure 5.4: Relation of computational time and number of nodes and factors in factor graph as a box plot.

100. It can be seen that the larger the number of nodes and factors in the graph the longer the computation time. This is approximately a linear relation, which indicates a linear complexity depending on the factor graph size.

In summary, it can be said that some statements can be made about the relationships between the input data and the complexity. While the resolution of the sensor has a linear influence on the number of neighborhood points and a quadratic influence on the total description calculation, in practice other characteristics of a sensor (in addition to the resolution) are decisive for the complexity. The selection's complexity depends heavily on the chosen parameters of the applied algorithms, the sensor resolutions, and the lengths of the descriptions. In addition, a linear relationship between the number of nodes and factors in the factor graph used for localization with respect to complexity can be identified.

Empirical Analyzes:

The empirical analyzes are based on a real-world data set, refer to Subsection 3.1.2. The code was implemented using C++. All calculations were performed on an Intel Core i7-6820HQ Central Processing Unit (CPU), or a NVIDIA Quadro M2000M Graphics Processing Unit (GPU), and with 16 gigabyte Random Access Memory (RAM).

For this, first, the computation time of the description calculation is determined. Here, real-world point clouds are recorded in a suburban area. Five minutes of the test drive, i.e., 3000 point clouds, are used to calculate descriptions for a sample geometry-based descriptor of the selected ones in Chapter 3, i.e., the FPFH descriptor. The FPFH descriptor is applied for practical reasons since there exists a sequential and parallel version in the Point Cloud Library (PCL) [107]. Therefore, the FPFH descriptor implementation of the PCL just needs to be integrated into this thesis' framework. The descriptions are determined for every point of each point cloud of the test drive with the FPFH descriptor. The average computation time of the descriptor calculation for the whole point cloud is calculated from all 3000 point clouds.

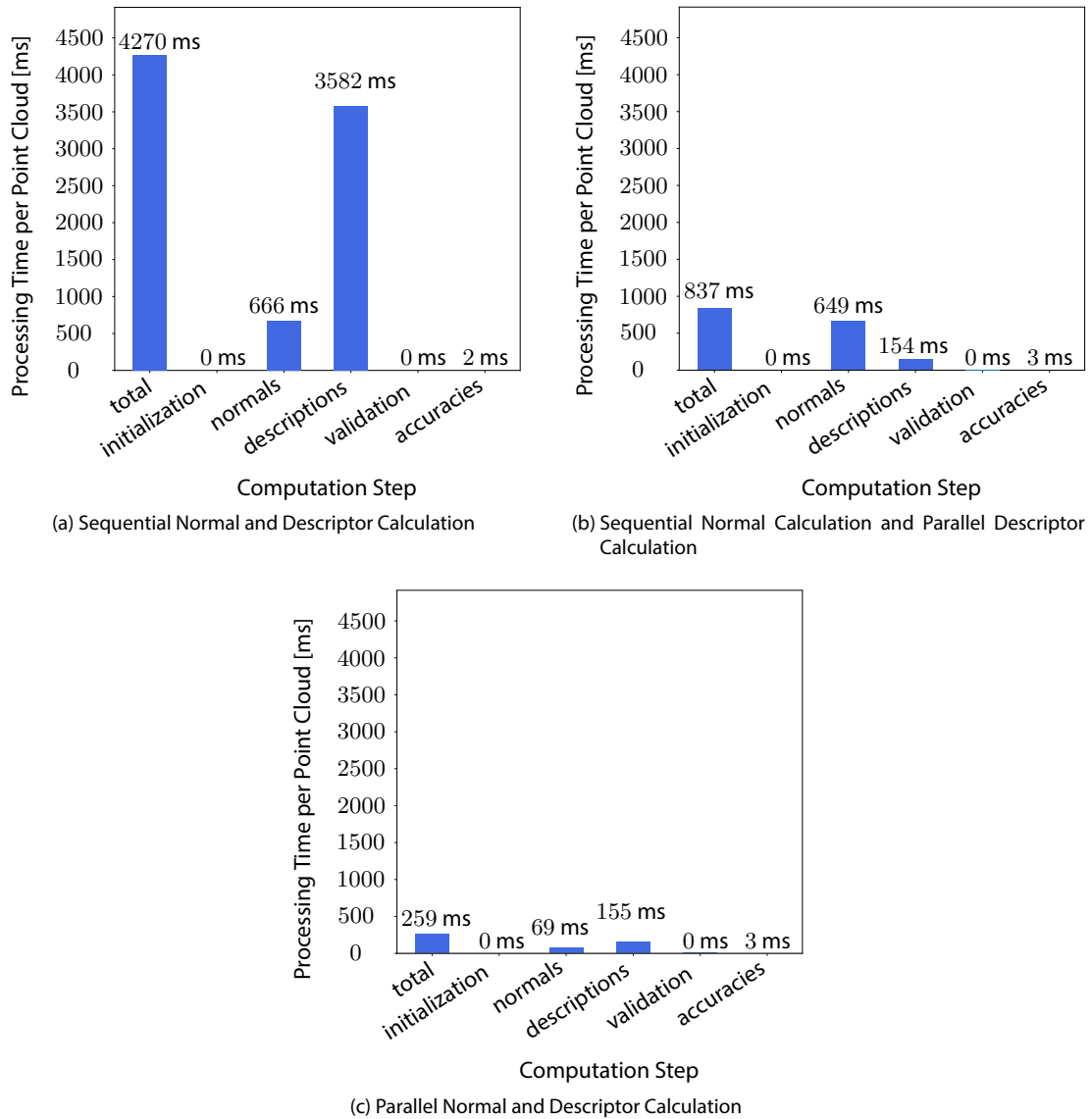


Figure 5.5: Comparison of computational times for sequential and parallel calculation of descriptions on the example of the FPFH descriptor. The average of 3000 point clouds is denoted.

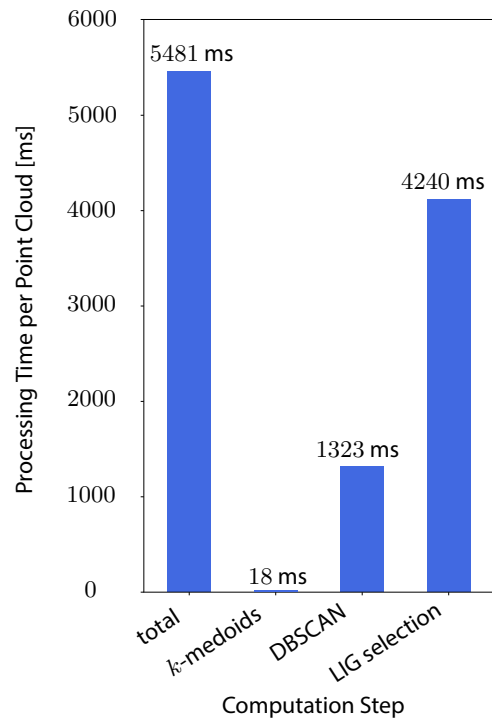


Figure 5.6: Demands on the computation effort of the OFAs detection on the example of the FPFH descriptor. The average of 2760 point clouds is denoted.

Figure 5.5 illustrates the results of these calculations. The required computation time is split into single determination steps. In this thesis' framework, an initialization phase is preceded to the PCL's normal and description calculation followed by an additional validation step and a determination of the accuracy of the calculation. That means the computation time of the normal and description calculation can be separately examined considering either the sequential or the parallel determination.

In Figure 5.5a, the normal and the description calculation are carried out sequentially. It can be seen that the main total calculation time of 4270 milliseconds arose from the actual description determination of 3582 milliseconds and the normal determination of 666 milliseconds. That means the description calculation constitutes 84% of the total computation time.

Computing the descriptions in parallel and the normals still sequentially, a total calculation time of 837 milliseconds is yielded, see Figure 5.5b. This is approximately 19% of the computation time compared to that of the previous determination. The description calculation amounts to 154 milliseconds, i.e., 4% of the sequential computation.

Figure 5.5c shows the components of the computation time when the normal and description calculation is executed in parallel. Here, a total time of 259 milliseconds is reached with 6% of the sequential computation time, which is much closer to a real-time capability for LiDAR sensors providing point clouds with 10 Hertz. The computation time of the parallel determination of the normals is 69 milliseconds, just 10% of the sequential determination.

Second, the computation time of the OFAs detection is computed. Similar to the experiment above but with a different data set, real-world point clouds are sampled in an urban area. 2760 point clouds are the input for the OFAs detection on the example of the FPFH descriptor. The OFAs are detected in every point cloud of the test drive. Detecting OFAs for every of the 2760 point clouds, the average computation time is determined.

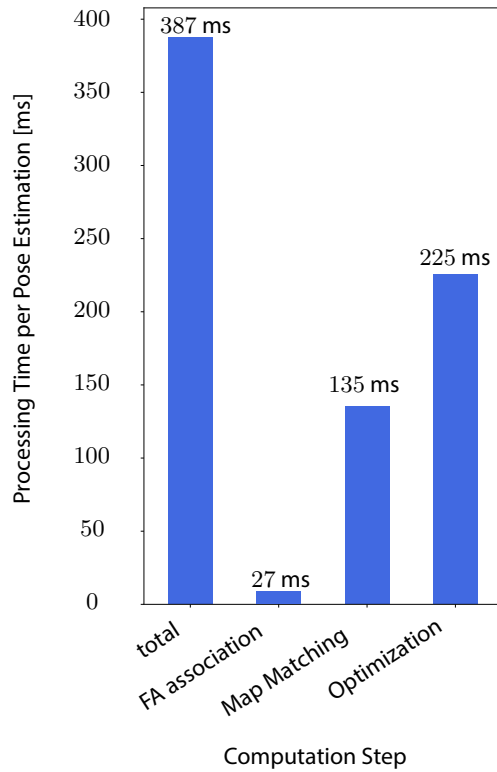


Figure 5.7: Demands on the computation effort of the actual localization on the example of the FPFH descriptor with a sliding window length of 100 vehicle poses and OFAs. The average of 2760 point clouds is denoted.

In Figure 5.6, the average computation time for the OFAs detection is shown. The computation time is divided into three calculation steps, after the k -medoids clustering with $k = 2$, after the DBSCAN clustering, and after the selection of the best OFAs according to the LIG value (LIG selection). The k -medoids is implemented as the Clustering LARge Applications (CLARA) algorithm by Kaufman and Rousseeuw [59]. It can be seen that the computation time in general is very large with 5481 milliseconds. While the k -medoid algorithm is relatively fast with 18 milliseconds, the DBSCAN and especially the selection of the best OFAs according to the LIG value are slow requiring 1323 milliseconds or 4140 milliseconds, respectively. On the one hand, this is due to the fact that the data is high-dimensional, which requires a lot of computation effort. On the other hand, especially the selection of the best OFAs was implemented straight forward without the intention of computational performance. As a solution to reduce these demands of computing resources, the selection of the best OFAs could be implemented with a focus on computational performance. Additionally, non-distinctive clusters could be sorted out after the k -medoid algorithm, i.e., before the DBSCAN clustering, refer to the definition of the distinctiveness in Section 4.3. These steps would reduce the computational time largely.

Third, the computation time of the actual localization is computed. The experiment is performed in the same environment as the second experiment on the computing resource, i.e., in an urban area with 2760 point clouds. The OFAs detected with the FPFH descriptor from the second experiment are the input for localization. MFAs were extracted from dense point clouds as described in the previous

chapter. The GPS data is recorded with a close-to-production NovAtel FlexPak-G2 OEMStar sensor. The odometry data from the EgoMaster modules was applied. The localization is performed with a sliding window length of 100 vehicle poses and OFAs, which is a crucial parameter for the amount of the computation time. A vehicle pose is calculated every 0.1 seconds.

Figure 5.7 visualizes the average computation time for the actual localization. The whole computation time is divided into three computation steps, Feature Area association, map matching, and the optimization of the graph, see Figure 5.1. The graph optimization is performed using g2o [67]. It can be seen that the computation time of localization of 387 milliseconds is smaller compared to the computation times of the other two steps of the *LiDAR-Feature-based Localization* (without parallelization). While the FA association is relatively fast with 27 milliseconds, the map matching and the optimization of the graph are slow requiring 135 milliseconds or 225 milliseconds, respectively. The actual optimization of the graph takes 20 milliseconds while building the graph requires 205 milliseconds. It can be seen that the graph itself is very large, i.e., contains many edges, cf. Figure 5.2. That is why the graph construction takes so long. The straight forward way to reduce the computation time is to choose a smaller sliding window length. Additionally, the map matching time could be decreased by performing the map matching just every few seconds not every cycle, in this case every 0.1 seconds. However, these procedures would reduce the accuracy of the estimated vehicle pose.

Concluding, the computation of the *LiDAR-Feature-based Localization* is very expensive. However, a parallelization and additional time reducing actions, e.g., the sorting of non-distinctive cluster before the spatial clustering, provide a high potential to reach real-time capability.

5.4. Summary

In this chapter, the FAs are integrated into a state-of-the-art localization algorithm. As algorithm a graph-based sliding window SLAM solution is picked.

The main task of the graph-based sliding window SLAM is to process the input data, i.e., OFAs, odometry data, MFAs, and GNSS data, robustly. This is done in two steps, see Section 5.2. As OFAs are collected for every on-board LiDAR point cloud, an association needs to be performed to determine which OFAs, extracted from the on-board point clouds of the current time step, can be matched with OFAs extracted from point clouds of former time steps. Therefore, the OFAs of all time steps are clustered based on their position and description data to find those OFAs which are likely the same.

After the association of OFAs, it is decided which of the OFA clusters correspond to which MFAs. This is called map matching. Again, their position and description data are taken into account to estimate the best rotation and translation between the two sets of OFA clusters and MFAs. The determination of map matches is performed for every time step independently. While repeating the map matching process once every time step, it yields various sets of map matches over time. Since these map matches may occur only once or a few times over time, they are not used in localization. Only those map matches are used where often enough OFA clusters are matched with the same MFA and the OFA clusters contain enough associated OFAs.

These map matches are then used to solve localization as an optimization problem estimating the optimal state vector, i.e., vehicle poses and OFA cluster representative positions. This is done in a numerical way applying the Gauss-Newton optimization algorithm.

While the concept of a non-semantic localization does not make assumptions about semantics in the environment it requires large computation effort. However, it can be shown that a parallelization enables a great reduction in computation time, demonstrating a potential real-time capability. The average accuracy reached with this non-semantic localization is approximately 40 centimeters and outperforms that of a GPS based localization.

6. Conclusion

In this thesis, the practicability of localization based on non-semantic elements detected in Light Detection And Ranging (LiDAR) data has been studied. This concept is called *LiDAR-Feature-based Localization*. Here, the key findings and contributions of this thesis are summarized, and promising directions of future work are indicated.

6.1. Key Findings and Contributions

The *LiDAR-Feature-based Localization* developed in this thesis consists of three steps: characterizing neighborhoods in LiDAR point clouds with descriptor algorithms, detecting significant and persistent non-semantic elements (called Feature Areas (FAs)) on-board and in the map, and computing an estimation of the position of the ego vehicle. The focus of this work is to investigate whether descriptor algorithms and elements extracted using them are suitable for localization, not to develop a fully integrated localization solution, which should also consider other sources of positioning information. These topics are presented in individual chapters, namely Chapters 3, 4, and 5.

For each topic, research questions have been defined based on a thorough literature review. Chapter 2 presents the findings of this review and the conclusions drawn from it. To summarize the conclusions, many related methods for each of the three steps can be found in the literature. However, it cannot always be said that they are applicable in the context of this thesis' concept.

First, the literature review revealed that, for a practical application, descriptor algorithms have not been studied thoroughly, yet. In most cases, the findings from the literature do not fully cover the properties of a descriptor algorithm needed to judge its suitability for localization. Many state-of-the-art descriptors are only evaluated with simulated data largely differing from a real world environment, such as uniformly sampled Computer-Aided Design (CAD) models. However, in practice, the environment is not captured in homogeneous, uniformly distributed point clouds. Additionally, a fixed number of neighbors or a fixed radius around the query point is set while determining descriptions for every query point of the point cloud. This is also not feasible for non-uniform point clouds. The artificial CAD models are artificial and often very different from objects occurring in street sceneries.

Second, in the reviewed literature, non-semantic elements are usually extracted based on their characteristics. This just takes the element's identifiability, but not their practicability for localization into account. For instance, some elements are selected based on their significantly large curvature and surface variation values. However, for localization, they have to be reliably detected under several variations, e.g., viewpoint changes, sensor noise, and inhomogeneous point densities.

These issues motivate several research questions. The key findings and contributions of each chapter to these research questions are listed below.

Are geometry-based descriptor algorithms well-suited for a real-world localization? To answer this research question, requirements for well-suited descriptor algorithms in the context of self-location of automated vehicles are defined in Chapter 3. They state that descriptor algorithms need to detect the local neighborhood structure expressively, that they need to differentiate between different object classes and classify similar objects in the same manner, independently of the point cloud densities. Following these requirements, state-of-the-art descriptor algorithms characterizing geometric data are evaluated thoroughly. For that, simulated data generated

with close-to-production-type LiDAR sensor models and real-world data are used. These experiments show that the descriptor algorithms meet many of the requirements, but lack distance independence and are highly dependent on the local neighborhood size.

Several methods and improvements are presented in Chapter 3 to resolve these deficiencies. This includes approaches on how to choose an optimal local neighborhood size and newly developed descriptor algorithms. In experiments, it is verified that the developed algorithms are insensitive to moderate changes in the distance.

Are intensity-based descriptor algorithms as useful for localization as geometry-based ones?

Chapter 3 presents an intensity-based descriptor algorithm developed in this thesis. Intensity-based descriptors only process one-dimensional intensity information from LiDAR point clouds compared to three-dimensional information processed by geometry-based descriptors. Calculations of odometry accuracies based on both types of descriptor algorithms indicate that they enable a comparable accuracy. Thus, intensity-based descriptor algorithms may be considered for realizing localization.

Which methods are well-suited to detect good Feature Areas? In a first step answering this question, criteria are defined in Chapter 4, specifying the requirements for the extraction of non-semantic elements, i.e., Feature Areas (FAs). They specify that i) areas rather than single key points should be detected, ii) the FAs should be detected in a non-semantic way, iii) the extraction method has to deal with descriptions determined from real-world data and not idealized synthetic data, and iv) the number of selected FAs must be flexibly adjustable to the available areas and needs of localization.

To meet these requirements a method based on clustering algorithms is defined, which fulfill these. Furthermore, these algorithms compress the used data while retaining most of the useful information. As the first step of this method, the geometry-based descriptors are calculated for those points of the point cloud with enough points in their local neighborhood. A clustering algorithm applied in the description space is performed to group points with similar descriptions using the k -medoid algorithm. These clusters of points with similar descriptions are the input for clustering in the Euclidean space. Here, each description clusters is grouped separately into multiple areas of spatially connected points using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

Examinations show that persistent and more accurate results than with existing key-point-based methods can be reached with this thesis' method. Additionally, a significant reduction in memory size can be accomplished by applying the proposed extraction method.

Which properties are well-suited to select good Feature Areas? Not every detected cluster provides information which is robust and useful for localization. This research question is answered in Chapter 4 by defining metrics for capturing the benefit of the FAs for localization. For that, a measure called *Localization Information Gain* (LIG) has been defined. It integrates several aspects, considering the distinctiveness, uniqueness, and spatial diversity of the extracted FAs. Those α percent of the computed clusters are kept which have the largest LIG value. This method's outcome are Feature Areas (FAs) in the map which can also be recognized in on-board data.

Map elements are called Map Feature Areas (MFAs), and detections in the test vehicle are called On-board Feature Areas (OFAs). The localization of this thesis is based on matching OFAs with MFAs. Therefore, as a first experiment, the LIG's suitability is evaluated performing a very simplistic localization algorithm. The results of this experiment demonstrate that the proposed extraction algorithm provides useful results.

How should Feature Areas be integrated into a state-of-the-art localization algorithm? This research question is answered in Chapter 5. Here, FAs have been integrated into a state-of-the-art localization algorithm, a graph-based sliding window solution. The main task to integrate FAs into localization is the association of data.

OFA are determined on-board for every LiDAR point cloud, i.e., for every time step. Thus, one object in the scenery gives rise to a number of OFAs in consecutive time steps. These are associated with each other by clustering them using their position and description data.

Subsequently, OFAs are related to MFAs is performed. This step is called map matching. In map matching, it is determined which of the OFA clusters correspond to which MFAs by comparing their position and description data. This yields the best rotation and translation of OFA clusters to the map for every time step. The matches may vary over time. Thus, only those map matches are included an optimal rotation and translation with these map matches minimizing overall deviations.

What is the practical benefit of the LiDAR-Feature-based Localization? The last research question aims at the core question of this thesis, the practicability of a localization based on non-semantic elements detected in LiDAR data. The answers to this research question are depicted in Chapters 4 and 5.

The precision obtained with an expensive non-semantic localization constructed from the developments of this thesis outperforms that of Global Positioning System (GPS) positioning, but does not meet the accuracy required for automated driving, yet. Enhancing the map by deleting artifacts and improving the implementation, e.g., the integration of polygonal chain OFAs into the factor graph, can reduce the position errors. However, it can be shown that the detection of non-semantic FAs ensures a scenery independent detection of elements in the environment, which is a great advantage and main motivation compared to semantic localization solutions.

It can be summarized that non-semantic elements provide scenery independent information for a localization, which, together with a robust implementation, enable the practicability of a non-semantic approach.

Even without including other positioning information the concept of this work achieves good localization results. This indicates that the integration of non-semantic elements into a full localization solution would yield a substantial localization. Consequently, this thesis achieves a fully positive answer to the specified research question, which guided the research of this work.

6.2. Future Work

In this thesis, a localization concept relying on non-semantic Feature Areas (FAs) detected in LiDAR data has been developed. It consists of describing points of point clouds, detecting significant and persistent FAs automatically on-board and offline in the map in a non-semantic way, and determining a vehicle pose based on these FAs. All of these steps were evaluated thoroughly. Nevertheless, several further issues should be addressed in consecutive research.

Keeping the Map Up To Date: An important task, which should be examined in future work, is the problem of keeping the map up to date. The localization which has been proposed in this thesis relies on a prerecorded map. If the environment changes after the map was generated, localization is affected. For instance, this could result in wrong map matches, like an on-board extracted FA is matched with an outdated map FA. This could induce a poor localization accuracy. As a result, updating the map is crucial for any practical application. One way to do that employs

the graph-based localization algorithm, which was introduced in Chapter 5. As the positions of the on-board detected FAs are included to the state vector of the vehicle which is optimized to solve the localization problem they are optimized as well. That means that those positions of the on-board detected FAs are calculated which explain all measurements best. These optimized on-board FAs positions could be used to update the map, i.e., delete, add, or correct the map elements. This contributes to the problem of dynamic objects in the map, refer to Subsection 4.5.3.

Hybrid Localization: What should also be investigated in future work is a hybrid localization relying on semantic and non-semantic elements, i.e., landmarks and FAs. This thesis only focuses on using non-semantic FAs for localization. As the computation of these elements is very expensive, a localization applying FAs, where few, a bad geometric constellation, or an ambiguous formation of semantic landmarks occur, should be examined. This is compatible with the graph-based localization applied in this thesis. In a first step, the landmarks and the FAs should be considered separately, i.e., the association and map matching should be carried out for landmarks and FAs individually. All valid matches of landmarks and FAs could then be fed into the optimization resulting in a localization depending on both types of elements.

Generate the Map from Online Data: The concept of this thesis has been to generate the map, i.e., MFAs, from dense LiDAR point clouds. To determine a vehicle pose, OFAs are matched with MFAs. OFAs are extracted from sparse on-board LiDAR point clouds. Experiments of Section 3.2 have shown that the descriptions from OFAs and MFAs differ. Hence, a map generated from globally referenced, optimized OFAs instead of MFAs might simplify the matching

AI-based Descriptor Algorithms: In this thesis, only handcrafted descriptors were applied to the concept of the *LiDAR-Feature-based Localization*. The outlook on Artificial Intelligence (AI)-based algorithms, refer to appendix A, considered two potential usages of AI techniques: one employing connected autoencoder, the other a siamese network. Neither techniques produced useful results in the experiments performed. They were not able to generalize when trained with the used test data. However, a more thorough investigation using a training data set, containing point clouds containing many different geometric objects, should be examined and could produce trained networks being able to generalize.

Reducing Computational Effort: As this thesis concentrated on the potential of *LiDAR-Feature-based Localization*, there has been no focus on keeping the computation time low. The analyzes of Subsection 5.3.2 show that the computation of the descriptions and the extraction of the non-semantic FAs take a lot of computation time. However, it is obvious that the parallelization of these algorithms would reduce computation time significantly. Hence, to parallelize and optimize the code is recommended for a real-time application of the *LiDAR-Feature-based Localization*.

Bibliography

- [1] Applanix. *Applanix POS LV 520 datasheet*, 2017.
- [2] P. Axelrad, C. J. Comp, and P. F. Macdoran. Snr-based multipath error correction for gps differential phase. *IEEE Transactions on Aerospace and Electronic Systems*, pages 650–660, 1996.
- [3] M. Baer, M. E. Bouzouraa, C. Demiral, U. Hofmann, S. Gies, and K. Diepold. Egomaster: A central ego motion estimation for driver assist systems. In *IEEE International Conference on Control and Automation*, pages 1708–1715, 2009.
- [4] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, pages 111–122, 1981.
- [5] T. D. Barfoot, C. McManus, S. Anderson, H. Dong, E. Beerepoot, C. H. Tong, P. Furgale, J. D. Gammell, and J. Enright. Into darkness: Visual navigation based on a lidar-intensity-image pipeline. In *Robotics Research*, pages 487–504, 2013.
- [6] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 404–417, 2006.
- [7] D. Belton and D. D. Lichti. Classification and segmentation of terrestrial laser scanner point clouds using local variance information. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, pages 44–49, 2006.
- [8] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 167–192, 1988.
- [9] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 239–256, 1992.
- [10] B. Bhanu, S. Lee, C. C. Ho, and T. Henderson. Range data processing: representation of surfaces by edges. In *International Conference on Pattern Recognition (ICPR)*, pages 236–238, 1986.
- [11] D. J. Bora and A. K. Gupta. Effect of different distance measures on the performance of k-means algorithm: An experimental study in matlab. (*IJCSIT*) *International Journal of Computer Science and Information Technologies*, 5:2501–2506, 2014.
- [12] M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *International Journal of Robotics Research*, pages 667–691, 2008.
- [13] M. Bosse and R. Zlot. Keypoint design and evaluation for place recognition in 2D lidar maps. *Robotics and Autonomous Systems*, pages 1211–1224, 2009.
- [14] C. Brenner. Vehicle localization using landmarks obtained by a lidar mobile mapping system. In *Proceedings of the Photogrammetric Computer Vision and Image Analysis*, pages 139–144, 2010.
- [15] S.-H. Cha and S. N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6):1355–1370, 2002.

- [16] H. Chen and B. Bhanu. 3D free-form object recognition in range images using local surface patches. In *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 136–139, 2004.
- [17] J. Chen and B. Chen. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, pages 223–236, 2008.
- [18] J. Collier, S. Se, V. Kotamraju, and P. Jasiobedzki. Real-time lidar-based place recognition using distinctive shape descriptors. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 2012.
- [19] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 603–619, 2002.
- [20] K. P. Cop, P. V. K. Borges, R. Dubé, K. P. Cop, P. V. K. Borges, and R. Dubé. DELIGHT: An efficient descriptor for global localisation using lidar intensities. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3653–3660, 2018.
- [21] M. Cummins and P. Newman. Highly scalable appearance-only slam - fab-map 2.0. *Proceedings of the Robotics: Science and Systems Conference (RSS)*, pages 1100–1123, 2009.
- [22] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.
- [23] M. Dawood, C. Cappelle, M. E. El Najjar, M. Khalil, and D. Pomorski. Harris, SIFT and SURF features comparison for vehicle localization based on virtual 3d model and camera. In *International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 307–312, 2012.
- [24] H. Deusch, J. Wiest, S. Reuter and D. Nuss, M. Fritzsche, and K. Dietmayer. Multi-sensor self-localization based on maximally stable extremal regions. In *IEEE Intelligent Vehicles Symposium*, pages 555–560, 2014.
- [25] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard. Rigid scene flow for 3d lidar scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1765–1770, 2016.
- [26] A. Dewan, T. Caselitz, and W. Burgard. Learning a local feature descriptor for 3D lidar scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4774–4780, 2018.
- [27] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, pages 112–122, 1973.
- [28] R. Dubé, A. Cramariuc, D. Dugas, J. I. Nieto, R. Siegwart, and C. Cadena. SegMap: 3D segment mapping using data-driven descriptors. *International Journal of Robotics Research*, 2018.
- [29] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.
- [30] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Fofou, and A. Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing.*, 2(3):267–279, 2014.

-
- [31] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong. 3D deep shape descriptor. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2319–2328, 2015.
- [32] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, pages 381–395, 1981.
- [33] A. Flint, A. Dick, and A. v. d. Hengel. Thrift: Local 3D structure recognition. In *DICTA: International Conference on Digital Image Computing: Techniques and Applications*, pages 182–188, 2007.
- [34] A. Flint, A. Dick, and A. van den Hengel. Local 3D structure recognition in range images. *IET Computer Vision*, pages 208–217, 2008.
- [35] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 224–237, 2004.
- [36] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, pages 32–40, 1975.
- [37] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25:130–150, 2006.
- [38] N. Gelfand and L. J. Guibas. Shape segmentation using local slippage analysis. In *Symposium on Geometry Processing*, pages 214–223, 2004.
- [39] E. Grilli, F. Menna, and F. Remondino. A review of point clouds segmentation and classification algorithms. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, pages 339–344, 2017.
- [40] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, pages 31–43, 2010.
- [41] H. Gross and U. Thoennessen. Extraction of lines from laser point clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, 36(3):86–91, 2006.
- [42] J. Guo, P. V. K. Borges, C. Park, and A. Gawel. Local descriptor for robust place recognition using LiDAR intensity. *Computing Research Repository*, pages 1470–1477, 2018.
- [43] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan. Rotational projection statistics for 3D local surface description and object recognition. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 105(1):63–86, 2013.
- [44] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan. TriSI: A distinctive local surface descriptor for 3D modeling and object recognition. In *Proceedings of the International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications*, pages 86–93, 2013.
- [45] Y. Guo, F. Sohel, M. Bennamoun, J. Wan, and M. Lu. RoPS: A local feature descriptor for 3D rigid objects based on rotational projection statistics. *International Conference on Communication, Signal Processing and Their Applications*, pages 1–6, 2013.
- [46] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan. 3D object recognition in cluttered scenes with local surface features: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.

- [47] X.-F. Han, J. Jin, J. Xie, M.-J. Wang, and W. Jiang. A comprehensive review of 3D point cloud descriptors. *Computing Research Repository*, 2018.
- [48] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of Alvey Vision Conference*, pages 147–151, 1988.
- [49] J.-H. Haunert and C. Brenner. Vehicle localization by matching triangulated point patterns. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 344–351, 2009.
- [50] M. Hebert, K. Ikeuchi, and H. Delingette. A spherical representation for recognition of free-form surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 681–690, 1995.
- [51] M. Himstedt, J. Frost, S. Hellbach, H. Böhme, and E. Maehle. Large scale place recognition in 2d lidar scans using geometrical landmark relations. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5030–5035, 2014.
- [52] C. Hungar, S. Brakemeier, S. Jürgens, and F. Köster. GRAIL: A gradients-of-intensities-based local descriptor for map-based localization using LiDAR sensors. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 4398–4403, 2019.
- [53] C. Hungar, J. Fricke, S. Jürgens, and F. Köster. Detection of feature areas for map-based localization using LiDAR descriptors. In *IEEE Workshop on Positioning, Navigation and Communications (WPNC)*, pages 1–6, 2019.
- [54] C. Hungar, F. Köster, and S. Jürgens. Ein Beitrag zur Karten-basierten Positionierung von Fahrzeugen mittels Mustererkennung in LiDAR-Daten. *AAET Automatisiertes und vernetztes Fahren*, pages 135–155, 2019.
- [55] C. Hungar, S. Jürgens, D. Wilbers, and F. Köster. Map-based localization with factor graphs for automated driving using non-semantic lidar features. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.
- [56] International Organization for Standardization. ISO 8855:2011: Road vehicles - vehicle dynamics and road-holding ability - vocabulary. Standard, DIN, November 2011.
- [57] Y. Ioannou, B. Taati, R. Harrap, and M. A. Greenspan. Difference of normals as a multi-scale operator in unorganized point clouds. *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 501–508, 2012.
- [58] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 433–449, 1999.
- [59] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.
- [60] G. Kim and A. Kim. Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809, 2016.
- [61] G. Kim, B. Park, and A. Kim. 1-day learning, 1-year localization: Long-term lidar localization using scan context image. *IEEE Robotics and Automation Letters*, pages 1948–1955, 2019.

-
- [62] R. Klokov and V. Lempitsky. Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 863–872, 2017.
- [63] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. V. Gool. Hough transform and 3D SURF for robust three dimensional classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 589–602, 2010.
- [64] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- [65] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
- [66] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, pages 498–519, 2001.
- [67] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, 2011.
- [68] V. Kumar, J. Chhabra, and D. Kumar. Performance evaluation of distance metrics in the clustering algorithms. *INFOCOMP Journal of Computer Science*, 13(1):38–52, 2014.
- [69] G. Lavoué, F. Dupont, and A. Baskurt. A new cad mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design*, pages 975–987, 2005.
- [70] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4372–4378, 2010.
- [71] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2007.
- [72] T. Li, H. Zhang, Z. Gao, Q. Chen, and X. Niu. High-accuracy positioning in urban environments using single-frequency multi-GNSS RTK/MEMS-IMU integration. *Remote Sensing*, 2018.
- [73] Y. Li and E. B. Olson. A general purpose feature extractor for light detection and ranging data. *Sensors*, 10(10):10356–10375, 2010.
- [74] Y. Li, X. Wu, Y. Chrysanthou, A. Sharf, D. Cohen-Or, and N. J. Mitra. GlobFit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, pages 52:1–52:12, 2011.
- [75] T. Litman. *Autonomous Vehicle Implementation Predictions: Implications for Transport Planning*. Victoria Transport Policy Institute, 2013.
- [76] N. Liu, Y. Yin, and H. Zhang. A fingerprint matching algorithm based on delaunay triangulation net. In *International Conference on Computer and Information Technology*, pages 591–595, 2005.
- [77] Waymo LLC. Introducing the 5th-generation waymo driver: Informed by experience, designed for scale, engineered to tackle more environments, 2020. URL <https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html>. [Access: 04/16/2020].
- [78] T.-W. R. Lo and J. P. Siebert. Local feature extraction and matching on range images: 2.5D SIFT. *Computer Vision and Image Understanding*, 113(12):1235–1250, 2009.

- [79] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.
- [80] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.
- [81] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song. L3-Net: Towards learning based LiDAR localization for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6382–6391, 2019.
- [82] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 281–297, 1967.
- [83] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Image and Vision Computing*, pages 761–767, 2002.
- [84] B. Matei, Y. Shan, H. S. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert. Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1111–1126, 2006.
- [85] C. Merfels. *Sensor fusion for localization of automated vehicles*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2018.
- [86] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010.
- [87] F. Mokhtarian, N. Khalili, and P. Yuen. Multi-scale free-form 3D object recognition using 3D models. *Image and Vision Computing*, 19:271–281, 2001.
- [88] F. Moosmann and C. Stiller. Velodyne SLAM. In *IEEE Intelligent Vehicles Symposium*, pages 393–398, 2011.
- [89] R. T. Ng and J. Han. CLARANS: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, pages 1003–1016, 2002.
- [90] A. Nguyen and B. Le. 3D point cloud segmentation: A survey. In *IEEE Robotics & Automation Magazine*, pages 225–230, 2013.
- [91] NovAtel. *NovAtel OEMStar*, 2015.
- [92] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22:281–289, 2003.
- [93] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the International Conference on Machine Learning*, pages 727–734, 2000.
- [94] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 406–413, 1955.
- [95] F. Poux, P. Hallot, R. Neuville, and R. Billen. Smart point cloud: Definition and remaining challenges. In *3D GeolInfo Conference*, pages 119–127, 2016.

-
- [96] S. Prakhya, B. Liu, and W. Lin. B-SHOT: A binary feature descriptor for fast and efficient keypoint matching on 3D point clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1929–1934, 2015.
- [97] S. Prakhya, J. Lin, V. Chandrasekhar, W. Lin, and B. Liu. 3DHoPD: A fast low-dimensional 3-D descriptor. *IEEE Robotics and Automation Letters*, pages 1472–1479, 2017.
- [98] S. Pu, G. Vosselman, and C. Vi. Automatic extraction of building features from terrestrial laser scanning. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, pages 33–39, 2006.
- [99] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view CNNs for object classification on 3d data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, 2016.
- [100] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.
- [101] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Conference on Neural Information Processing Systems*, pages 5099–5108, 2017.
- [102] T. Rabbani, F. A. Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraint. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, pages 248–253, 2006.
- [103] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, pages 244–256, 1972.
- [104] T. Reid, S. Houts, R. Cammarata, G. Mills, S. Agarwal, A. Vora, and G. Pandey. Localization requirements for autonomous vehicles. *Computing Research Repository*, 2019.
- [105] Riegl. *Riegl VQ-450 Datasheet*. Riegl Laser Measurement Systems, 2013.
- [106] G. Riegler, A. O. Ulusoy, and A. Geiger. OctNet: Learning deep 3D representations at high resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6620–6629, 2017.
- [107] R. B. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, 2011.
- [108] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5117–5122, 2008.
- [109] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz. Persistent point feature histograms for 3D point clouds. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, 2008.
- [110] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, 2009.

- [111] S. Salti, F. Tombari, and L. Di Stefano. A performance evaluation of 3D keypoint detectors. In *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 236–243, 2011.
- [112] S. Salti, F. Tombari, L. Di Stefano, S. Salti, F. Tombari, and L. Di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125: 251–264, 2014.
- [113] A. D. Sappa and M. Devy. Fast range image segmentation by an edge detection strategy. In *IEEE 3-D Digital Imaging and Modeling*, pages 292–299, 2001.
- [114] U. Scheunert, H. Cramer, and G. Wanielik. Precise vehicle localization using multiple sensors and natural landmarks. *Proceedings of the International Conference on Information Fusion*, pages 649–656, 2004.
- [115] A. Schlichting and C. Brenner. Localization using automotive laser scanners and local pattern matching. In *IEEE Intelligent Vehicles Symposium*, pages 414–419, 2014.
- [116] A. Schlichting and U. Feuerhake. Global vehicle localization by sequence analysis using lidar features derived by an autoencoder. In *IEEE Intelligent Vehicles Symposium*, pages 656–661, 2018.
- [117] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, pages 214–226, 2007.
- [118] F. Schuster, M. Wörner, C. G. Keller, M. Haueis, and C. Curio. Robust localization based on radar signal clustering. In *IEEE Intelligent Vehicles Symposium*, pages 839–844, 2016.
- [119] J. Serafin, E. Olson, and G. Grisetti. Fast and robust 3D feature extraction from sparse point clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4105–4112, 2016.
- [120] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27: 379–423, 1948.
- [121] I. Sipiran and B. Bustos. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, pages 963–976, 2011.
- [122] L. Smith. *A Tutorial On Principal Components Analysis*, 2002.
- [123] R. Spangenberg, D. Goehring, and R. Rojas. Pole-based localization for autonomous vehicles in urban scenarios. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2161–2166, 2016.
- [124] Statistisches Bundesamt. Unfallentwicklung auf Deutschen Strassen 2017, 2017.
- [125] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Point feature extraction on 3D range scans taking into account object boundaries. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2601–2608, 2011.
- [126] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, 2015.
- [127] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, pages 1383–1392, 2009.

-
- [128] B. Taati and M. Greenspan. Satellite pose acquisition and tracking with variable dimensional local shape descriptors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4–9, 2005.
- [129] B. Taati, M. Bondy, P. Jasiobedzki, and M. Greenspan. Variable dimensional local shape descriptors for object recognition in range data. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.
- [130] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer. Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, pages 407–412, 2007.
- [131] S. Thrun and J. J. Leonard. *Springer Handbook of Robotics*, chapter Simultaneous Localization and Mapping, pages 871–889. Springer Berlin Heidelberg, 2008.
- [132] E. Tola, V. Lepetit, and P. Fua. DAISY: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830, 2010.
- [133] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.
- [134] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 356–369, 2010.
- [135] F. Tombari, S. Salti, and L. Di Stefano. Unique shape context for 3D data description. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 57–62, 2010.
- [136] F. Tombari, S. Salti, and L. Di Stefano. A combined texture-shape descriptor for enhanced 3D feature matching. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 809–812, 2011.
- [137] F. Tombari, S. Salti, and L. Di Stefano. Performance evaluation of 3D keypoint detectors. *International Journal of Computer Vision*, 102(1-3):198–220, 2013.
- [138] M. Tomono. A scan matching method using euclidean invariant signature for global localization and map building. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 866–871, 2004.
- [139] D. Tóvári and N. Pfeifer. Segmentation based robust interpolation- a new approach to laser data filtering. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, pages 79–84, 2005.
- [140] B. R. Townsend and P. C. Fenton. A practical approach to the reduction of pseudorange multipath errors in a L1 GPS receiver. *International Technical Meeting Satellite Division Institute Navigation ION GPS*, pages 143–148, 1994.
- [141] B. R. Townsend, D. J. R. van Nee, P. C. Fenton, and K. J. van Dierendock. Performance evaluation of the multipath estimating delay lock loop. *International Technical Meeting Satellite Division Institute Navigation ION GPS*, pages 503–514, 1995.
- [142] Trimble. *Trimble MX8 Datasheet*. Trimble Navigation Limited, 2013.
- [143] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

- [144] Uber Advanced Technologies Group. Introducing uber atg's latest self-driving vehicle, 2020. URL <https://www.uber.com/us/en/careers/teams/advanced-technologies-group/>. [Access: 04/16/2020].
- [145] R. Unnikrishnan and M. Hebert. Multi-scale interest regions from unorganized point clouds. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8, 2008.
- [146] M. A. Uy and G. H. Lee. PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Computing Research Repository*, pages 4470–4479, 2018.
- [147] T. Vaa, M. Penttinen, and I. Spyropoulou. Intelligent transport systems and effects on road traffic accidents: State of the art. *Intelligent Transport Systems (IET)*, pages 81–88, 2007.
- [148] C. Valgren and A. J. Lilienthal. SIFT, SURF and seasons: Long-term outdoor localization using local features. In *Proceedings of the European Conference of Mobile Robots (ECMR)*, pages 253–258, 2007.
- [149] N. Vandapel, D. F. Huber, A. Kapuria, and M. Hebert. Natural terrain classification using 3-D ladar data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5117–5122, 2004.
- [150] Velodyne. *VLP-32 User Manual*. Velodyne LiDAR, Inc., 63-9325 rev. B edition, 2017.
- [151] Velodyne. *VLP-16 User Manual*. Velodyne LiDAR, Inc., 2018.
- [152] I. A. Venkatkumar and S. J. K. Shardaben. Comparative study of data mining clustering algorithms. In *International Conference on Data Science and Engineering (ICDSE)*, pages 1–7, 2016.
- [153] A.-V. Vo, L. Truong-Hong, D. Laefer, and M. Bertolotto. Classification and segmentation of terrestrial laser scanner point clouds using local variance information. *ISPRS International Journal of Photogrammetry and Remote Sensing*, pages 88–100, 2015.
- [154] Volkswagen AG. Autonomous driving – on the way to market maturity, 2020. URL <https://www.volkswagenag.com/en/news/stories/2019/11/autonomous-driving-on-the-way-to-market-maturity.html>. [Access: 05/20/2020].
- [155] G. Vosselman, B. Gorte, G. Sithole, and T. Rabbani. Recognising structure in laser scanner point clouds. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information (ISPRS)*, pages 33–38, 2004.
- [156] M. A. Wani and H. R. Arabnia. Parallel edge-region-based segmentation algorithm targeted at reconfigurable multiring network. *The Journal of Supercomputing*, pages 43–62, 2003.
- [157] G. Weiss, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 595–601, 1994.
- [158] T. Weiss, N. Kaempchen, and K. Dietmayer. Precise ego-localization in urban areas using laser-scanner and high accuracy feature maps. In *IEEE Intelligent Vehicles Symposium*, pages 284–289, 2005.
- [159] J. Wiest, H. Deusch, D. Nuss, S. Reuter, M. Fritzsche, and K. C. J. Dietmayer. Localization based on region descriptors in grid maps. *IEEE Intelligent Vehicles Symposium*, pages 793–799, 2014.

-
- [160] D. Wilbers, C. Merfels, and C. Stachniss. Localization with sliding window factor graphs on third-party maps for automated driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5951–5957, 2019.
- [161] R. Wolcott and R. Eustice. Fast LIDAR localization using multiresolution Gaussian mixture maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2814–2821, 2015.
- [162] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.
- [163] S. M. Yamany and A. A. Farag. Surface signatures: an orientation independent free-form surface representation scheme for the purpose of objects registration and matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1105–1120, 2002.
- [164] H. Yamauchi, S. Lee, Y. Lee, Y. Ohtake, A. Belyaev, and H.-P. Seidel. Feature sensitive mesh segmentation with mean shift. In *Shape Modeling International (SMI)*, pages 236–243, 2005.
- [165] Z. J. Yew and G. H. Lee. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–646, 2018.
- [166] D. Yin, Q. Zhang, J. Liu, X. Liang, Y. Wang, J. Maanpää, H. Ma, J. Hyypä, and R. Chen. CAE-LO: Lidar odometry leveraging fully unsupervised convolutional auto-encoder for interest point detection and feature description. *arXiv preprint arXiv:2001.01354*, 2020.
- [167] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. Surface feature detection and description with applications to mesh matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 373–380, 2009.
- [168] J. Zhang and S. Singh. LOAM: lidar odometry and mapping in real-time. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2014.
- [169] Y. Zhong. Intrinsic shape signatures: A shape descriptor for 3D object recognition. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 689–696, 2009.

List of Figures

1.1. Concept of the <i>LiDAR-Feature-based Localization</i>	4
1.2. Overview of the thesis' central part	9
2.1. Overview of 3D feature descriptor types with some examples	12
2.2. Darboux frame	14
2.3. Relations between sampling points for the description calculation of the Point Feature Histogram (PFH) and Fast Point Feature Histogram (FPFH)	15
2.4. Local neighborhood partitions of different description calculations	17
2.5. Scheme of Rotational Projection of Statistics (RoPS) description construction for one sample rotation	18
2.6. Specification of the simplex angle	19
2.7. Overview of extraction of non-semantic elements from 3D point clouds with some examples.	20
2.8. Illustration of 3D HT	25
2.9. Illustration of RANSAC	26
2.10. The three steps of the k -means algorithm	27
2.11. Overview of the localization methods based on non-semantic 3D LiDAR data with some examples.	28
2.12. Visualization of the Scan Context descriptor algorithm	29
2.13. The architecture of the AI-based localization with local features by Lu et al.	31
2.14. The three coordinate systems employed in this thesis	36
2.15. Structure of the considered pixels for the Daisy description calculation	40
2.16. The coordinates of the considered pixels for the calculation of the Daisy descriptor	41
2.17. The three point categories used in the DBSCAN algorithm	43
2.18. The three steps of the k -medoid algorithm	44
2.19. Representation of a sample factor graph	48
3.1. Sensor setup, architecture, and image of the test vehicles	56
3.2. Sample LiDAR point clouds of each data set type used in this thesis	57
3.3. Eigenvalues of a synthetic vertical plane as a function of the radial local neighborhood radius	61
3.4. The ratio of points in a histogram bin as a function of the local neighborhood radius	62
3.5. Normalized Euclidean distance as a function of the local neighborhood radius in a logarithmic scale for every angle of the FPFH description using the example of a dense real-data wall	63
3.6. Description deviations of a rotated edge compared to a non-rotated edge as a function of the viewing angle	69
3.7. Description deviations of a sparse, synthetic pillar compared to a dense, synthetic pillar as a function of the distance to the pillar	70
3.8. Description deviations of synthetic edge points depending on the distance to the exact edge	72
3.9. Description deviations of real edge points depending on the distance to the exact edge	73

3.10. Normalized Euclidean distance between two descriptions computed with local neighborhoods with increasing radii as a function of the local neighborhood radius	76
3.11. The ratio of points in a histogram bin as a function of the local neighborhood radius	77
3.12. Illustration of two angles of incidence $\alpha_{1,2}$	78
3.13. Transformation from the Sensor Reference Frame (SRF) into the Local Reference Frame (LRF)	79
3.14. 2D Grid variants	79
3.15. 3D Grid variants	80
3.16. Grid points of the Daisy descriptor	82
3.17. Description deviations of a sparse pillar compared to a dense pillar as a function of the distance	86
3.18. Scheme of the proposed GRAdients of Intensities as a Local descriptor (GRAIL) descriptor	89
4.1. Scheme of the proposed FA generation method	102
4.2. Visualization of the division for the spatial diversity	106
4.3. Illustration of the on-board Localization Information Gain (LIG) value	108
4.4. Sample OFAs and MFAs	110
4.5. Sample 2D representations of MFAs and OFAs	111
4.6. Cluster representatives of synthetic and real-world wall	113
4.7. Structure and content of the map data format	117
4.8. Test trajectory	119
4.9. Sample MFAs of five different sceneries	121
4.10. Comparison of the memory size of the raw data map and FAs map	122
5.1. Scheme of one time step of the localization computation	132
5.2. Sample factor graph in a real-world data set	138
5.3. Computed vehicle trajectory compared to a reference and a GPS trajectory	143
5.4. Relation of computational time and size of factor graph	148
5.5. Comparison of computational times for calculation of descriptions	149
5.6. Demands on the computation effort of the OFAs detection	150
5.7. Demands on the computation effort of the actual localization	151
A.1. Local neighborhood partitions of AI-based descriptors	177
A.2. Architecture of each of the two connected autoencoders	179
A.3. Value of additional penalty for different weight values on the validation set	180
A.4. Clustering the descriptions of points	181
A.5. Architecture of the siamese network	182
A.6. Architecture of the fully connected network	182
A.7. Architecture of the dense blocks	182
A.8. Receiver Operating Characteristic (ROC) curves using the test data set	184
A.9. ROC curves using the additional point cloud	184
B.1. Clustering of a sample point cloud	188
C.1. Distinctive shapes of the intensity patterns the GRAIL descriptor can differentiate between.	190
C.2. Least squares regression of reflectivity returns and distance to highly reflective material	190
C.3. Sample key points extracted with selection method for the GRAIL descriptor in a dense map and sparse on-board point cloud	192

List of Tables

2.1. Parameters of the Daisy description calculation - based on [132].	39
3.1. Distance matrix of FPFH, Signatures of Histograms of Orientations (SHOT), and RoPS medoid descriptions	66
3.2. (Normalized) Euclidean distances between sparse and dense real data for the descriptors FPFH, SHOT, and RoPS	67
3.3. Standard deviations measuring the allocation accuracy of an edge point for FPFH, SHOT, and RoPS	71
3.4. Daisy parameters of all three grid variants	82
3.5. Standard deviations for the smoothing of the orientation maps	83
3.6. (Normalized) Euclidean distances between real-data corner and pillar	88
3.7. Average RMS and standard deviation of absolute and rotational odometry errors of the GRAIL descriptor	92
3.8. Average RMS and standard deviation of absolute and rotational odometry errors of the GRAIL descriptor	92
4.1. Evaluation of clustering algorithms for clustering in the description space	112
4.2. Evaluation of clustering algorithms for clustering in the Euclidean space	114
4.3. Average 2D RMS position errors and standard deviations determined with the proposed extraction method	119
5.1. Evaluation of localization algorithms	129
5.2. Absolute RMS position errors and standard deviation of the <i>LiDAR-Feature-based Localization</i>	144
5.3. (Average) Total number of points within the point clouds, (average) number of points within the preprocessed point clouds, and average number of points within the local neighborhoods for description calculation	146
A.1. Different architectures and input-output combinations	178
A.2. Accuracy of the network	183
A.3. Validation accuracies of the description clustering method for autoencoder one	185
A.4. Validation accuracies of the description clustering method for autoencoder two	186
C.1. Distance metric of twelve distinctive shapes of GRAIL in percent	191
C.2. Description differences of minor changes of shapes of GRAIL in percent	191

Appendices

A. AI-Based Descriptors

This thesis focuses on the application of hand-crafted descriptors, such as the FPFH, RoPS, SHOT algorithms. Since it was demonstrated that AI is an effective method in several activities, in this chapter, two novel learned descriptors are designed for the task of this thesis. The first AI-based descriptor is based on autoencoders. The second AI-based descriptor follows the idea of a siamese network. Both descriptors rely on preprocessed LiDAR data. Therefore, at first, the preprocessing is introduced followed by a detailed explanation of the two descriptor algorithms.

A.1. Preprocessing of AI-Based Descriptors

To be able to determine descriptions using a neural network, it is important to transform the neighborhood so that the resulting representation has a fixed dimension. This problem is solved by using a voxel grid, i. e. a regular grid in the three-dimensional Euclidean space.

Let $\mathbf{p}_j^{x,y,z} \in \mathcal{P}^{x,y,z}$ be a point of a point cloud and let $r \in \mathbb{R}$ be a fixed radius. First, the local neighborhood is computed as follows:

$$\mathcal{N}(\mathbf{p}_j^{x,y,z}) = \{(\mathbf{p}^{x,y,z}, \mathbf{p}^i) \in \mathcal{P} = (\mathcal{P}^{x,y,z}, \mathcal{P}^i) \mid \|\mathbf{p}^{x,y,z} - \mathbf{p}_j^{x,y,z}\|_\infty \leq r\}. \quad (\text{A.1})$$

All points allocated inside a cube with center $\mathbf{p}^{x,y,z}$ and edge length $2r$ are of the local neighborhood $\mathcal{N}(\mathbf{p}_j^{x,y,z})$. The local neighborhood is partitioned in multiple rectangular boxes, called voxels, for example three along each of the three spatial axes. See Figure A.1 for this example of the local neighborhood partitions.

In this case, a fixed local neighborhood radius r is applied. A point's description is only calculated if the local neighborhood radius is ideal, refer to definition 3.5. Additional points are rejected if the local neighborhood includes fewer than 30 points. Otherwise, the point's description might not be representative of the local neighborhood, especially when applying the partitioned cube.

A Principal Component Analysis (PCA) is implemented as LRF after determining the local neighborhood, in order to ensure the independence of the viewing angle, cf. requirement 3.4 (v). In this way,

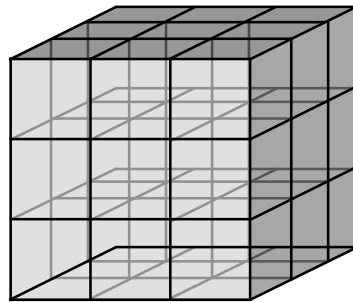


Figure A.1: Local neighborhood partitions of AI-based descriptors

Table A.1: Different architectures and input-output combinations. The \rightarrow denotes which type of input is mapped to which type of output.

architecture name	first autoencoder	second autoencoder	description size
ssdd2	sparse \rightarrow sparse	dense \rightarrow dense	$4 \times 4 \times 2$
ssdd4	sparse \rightarrow sparse	dense \rightarrow dense	$4 \times 4 \times 4$
sdds2	sparse \rightarrow dense	dense \rightarrow sparse	$4 \times 4 \times 2$
sdds4	sparse \rightarrow dense	dense \rightarrow sparse	$4 \times 4 \times 4$

the local neighborhood is transformed into a viewing angle independent coordinate system. This is done with a linear orthogonal transformation, resulting in a coordinate system where the largest variance of the data points is on the x -axis, the second largest variance on the y -axis, and the smallest variance on the z -axis.

Since a cube is not rotationally symmetric some points of the local neighborhood, prior to applying the PCA, no longer are part of the transformed local neighborhood and the other way around. Therefore, before the transformation, a local neighborhood radius $\sqrt{2}r$ is considered. After the transformation, the local neighborhood radius is reduced to r .

Then, in each voxel, the mean squared distance to the center $\mathbf{p}_j^{x,y,z}$ and the mean reflectivity of points within the voxel are stored. If a subset is empty both values are set to 0.

This voxel grid is the input for both AI-based descriptors, which are explained in the following.

A.2. Descriptor Based on Connected Autoencoders

In the following section, a method of learning a descriptor is presented using autoencoders.

The network of autoencoders consist of many layers which are divided into two parts. First, autoencoders encode high-dimensional input data to a latent space representation. Then, the decoder reconstructs the input from the latent space representation.

In this approach, the autoencoder architecture is applied so that the network is forced to learn the most important information of the input by encoding the input data, i. e. the voxel grid of size $16 \times 16 \times 2$. The output at the bottleneck layer can be interpreted as the input's description since it contains compressed information of the input data.

The idea of applying autoencoders is to enforce similarity of points' descriptions of a dense and a sparse point cloud of similar objects, i. e. learning requirement 3.4 (iv). This is realized with two connected autoencoders, which have the same architecture. The first autoencoder is supposed to learn encoding and decoding of dense voxels and the second of sparse voxels. The two autoencoders run in parallel and are connected at their bottleneck layers by a penalty term that penalizes dissimilar outputs. The connected autoencoders represent two descriptor-functions which generate descriptions from sparse and dense data, respectively.

Figure A.2 illustrates the architecture of each autoencoder of the two connected autoencoders. The convolutional kernels are of size 3×3 with padding so that the spatial dimension stays the same. A stride length of 2 is applied, which reduces the spatial dimension by a factor of two. In each layer except the last, ReLU is implemented as activation function. The activation function of the last layer has to be selected suitable for the values of the voxel being reconstructed. Therefore, an activation function with outputs in the range $[-1, 0]$ is needed, refer to Section A.1. Hence, tanh is applied here.

In order to evaluate the connected autoencoder approach, four variations of the architecture are compared to one another. They are summarized in Table A.1. Additional to learning a reconstruction of the input, it is examined if learning a mapping from sparse to dense voxels and dense to sparse

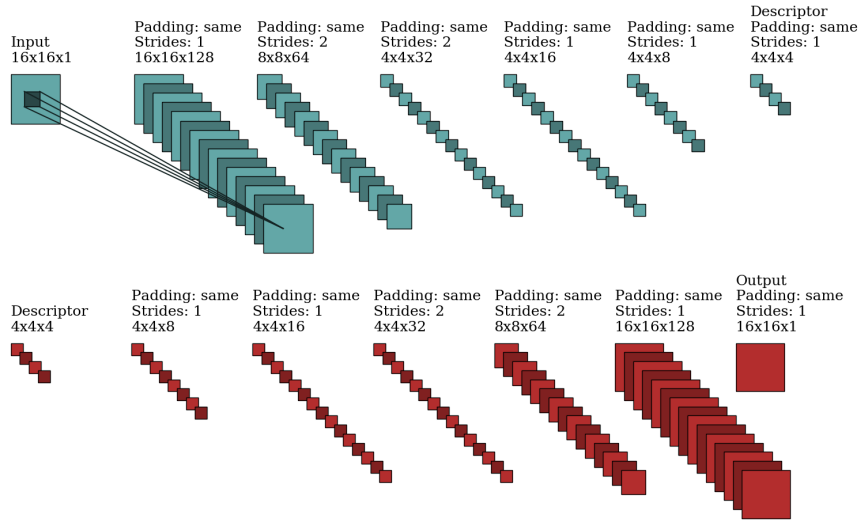


Figure A.2: Architecture of each of the two connected autoencoders

patches leads to reasonable results. Furthermore, a large ($4 \times 4 \times 4$) and a small ($4 \times 4 \times 2$) description size are compared.

All of these different architectures are trained with synthetic and real-world sparse and dense training data, refer to Subsection 3.1.2. Therefore, synthetic objects like planes, spheres and cylinders in different sizes are constructed. Transforming and combining them results in a complete scene with multiple objects. The voxel grids of these synthetic scenes and real-world point clouds are used as training data. The goal of the training is to solve the reconstruction task and minimize the difference of the descriptions of the same point. The binary cross entropy function comparing input and output data is used to evaluate the network parameters with an added weighted squared error penalty term comparing the two descriptions of the connected autoencoders. As optimizer, Adam is applied. Adam is an upgrade of the stochastic gradient descent, as it uses various learning rates for each network parameter and adapts them in the training process.

Now, the trained network is evaluated to check requirements 3.4 (iii) and (iv). It is investigated if the two descriptions determined by the network are similar when a penalty is enforced for dissimilar descriptions. In addition, it is examined if the determined descriptions can be applied to distinguish between dissimilar objects by clustering the descriptions of one scene with different objects.

The similarity of dense and sparse descriptions is examined by checking the value of the added penalty term of the loss function in relation to the added weight. For the evaluation of the network, the real-world data is used, comprising 8267 voxel pairs. The results are illustrated in Figure A.3, where the third architecture of Table A.1 was trained on 80% of the data set. It can be seen that with larger values of the penalty weight the penalty itself decreases, which is to be expected. That means, the more the similarity between the descriptions of the two autoencoders is forced the smaller the penalty term gets.

In order to examine if the determined descriptions can be applied to distinguish between dissimilar objects, a clustering in the descriptor space is applied. Therefore, in the second experiment, the descriptions are grouped into clusters and evaluate if, for example, walls are grouped into a different clusters than poles. For this purpose, the knowledge about which patch was generated from which type of object is used by applying the synthetic data set. More precisely, the data used for creating the

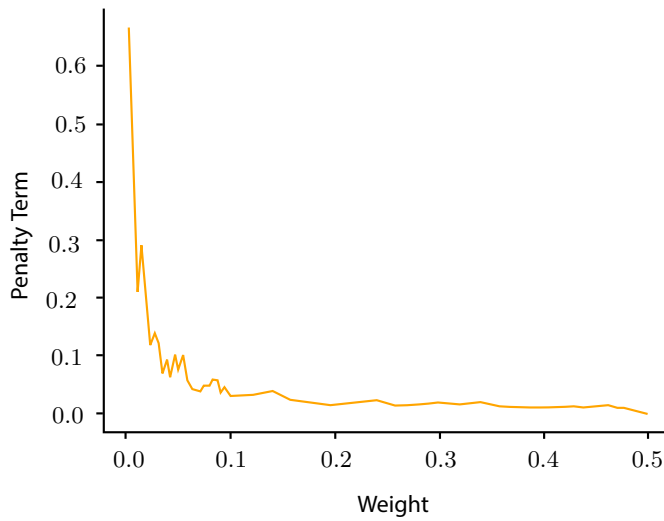


Figure A.3: Value of additional penalty for different weight values on the validation set

voxels comprises 562 sparse synthetic point clouds of poles and 542 sparse synthetic point clouds of walls with varying distances from the sensor. From those, 4328 voxel pairs of poles and 14054 voxel pairs of walls are determined. As clustering algorithm the k -medoids with $k = 2$ is used, refer to Sub-section 4.4.1. The clustering is performed on the four different architectures as shown in Table A.1. Furthermore, 46 different weight parameters of the additional penalty term between 0.0 and 0.5 are applied. In total, this yields to 368 different parameter combinations being investigated. Each network is trained for a minimum of 25 epochs and a maximum of 40 epochs. Each epoch takes at least 5 minutes to train. With the time for evaluation in between epochs, a total training time of more than 1000 hours is gained.

After each epoch, the results of the clustering are reported for the training (80% of the dataset) and the test (20% of the dataset) set, respectively. They are shown in Table A.3 and Table A.4. The results show that it is not clear which parameters provide the best results because many different combinations lead to similar results. Using a bias, seems to hinder the learning of the network. There seems to be no significant difference in using a description size of $4 \times 4 \times 2$ or $4 \times 4 \times 4$.

For an illustration of the results, the descriptions of each point in a scene are computed and visualized in Figure A.4. For this purpose, a pretrained network is applied. In Figure A.4, the results with the third architecture of Table A.1 with a penalty weight of 0.024 are shown. It can be seen that the points are not clustered according to their object class, rather the network seems to detect edges. This could be due to the fact that the voxels might contain redundant and misleading information or the chosen parameters might not be optimal.

Concluding, the connected autoencoder succeeds in producing similar descriptions for sparse and dense voxel pairs on real data. The second experiment has shown that the network cannot distinguish between dissimilar objects but rather detects edges in the voxels.

A.3. Descriptor Based on Siamese Network

This section presents an approach of learning a descriptor using a siamese network with the idea of obtaining similar descriptions for points lying on objects with comparable geometric structures.

The goal is to train the network with two voxel grids of two local neighborhoods of size $16 \times 16 \times 2$

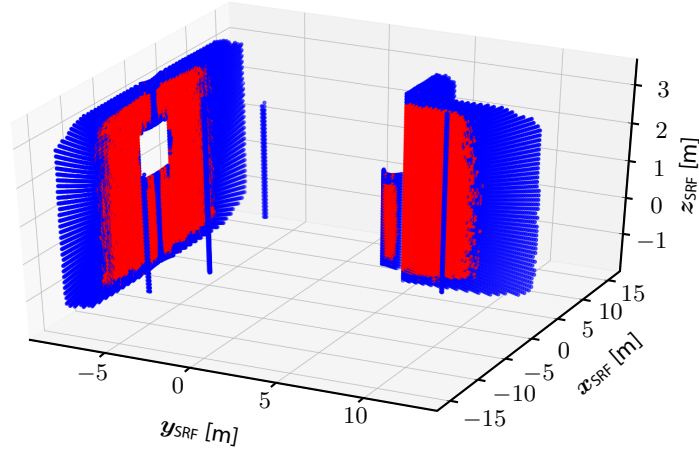


Figure A.4: Clustering the descriptions of points of a synthetically generated scene with walls and poles.

as input, which determines descriptions for each voxel grid and categorizes the descriptions as either a match or a non-match. The idea of this network is based on the work of Dewan et al. [26]. They use the rigid scene flow algorithm [25] to classify two points in consecutive point clouds to define whether they belong to the same object, a match, or not, a non-match. This method may result in wrong labels. For example, a point cloud might contain walls on each side of the street. Here, the rigid scene flow algorithm labels points of the same wall as a match and points of the different walls as a non-match. Thus, this method does not ensure that descriptions belonging to similar geometric structures are similar.

For this purpose, in this work, a different labeling process is developed. Here, the idea is to label data, i. e. voxels, that are categorized according to the geometric structure of the underlying object. Therefore, the FPFH descriptor computing descriptions based on the curvature within a local neighborhood is applied. The FPFH descriptions are similar if the points belong to objects with similar curvatures. This enables the labeling of the data with the FPFH descriptor. However, other descriptors may also be applied.

For labeling the data, real-world LiDAR point clouds sampled with on-board sensors are the input for the determination of the FPFH descriptions. These descriptions are clustered into two classes with the k -medoid algorithm, refer to Subsection 4.4.1. As a result, the clustering generates labeled point clouds where two points either belong the same class, called match, or to the other class, called non-match.

The labeled data is applied in a siamese network, whose architecture is illustrated in Figure A.5. Siamese networks are neural networks with two identical networks sharing the same weights and running in parallel. The network learns the descriptions and a metric for classifying the descriptions as match or non-match. The number of channels of the network are varied depending on the parameter $s \in \mathbb{R}_{>0}$. The input of the network are two $16 \times 16 \times 2$ voxel grids. Each stream of the siamese network is similar to an encoder of an autoencoder. That means that the input voxel grid are transformed to a lower-dimensional space with two convolution layers which are followed by an average pooling layer, two dense blocks, again followed by average pooling layers, and a bottleneck layer, i. e. which is a convolution layer. After every convolution layer follows an application of the ReLU activation function, except in the dense blocks. The last convolution layer outputs the learned description.

The architecture of the dense blocks is visualized in Figure A.7. Here, the input of the dense block is

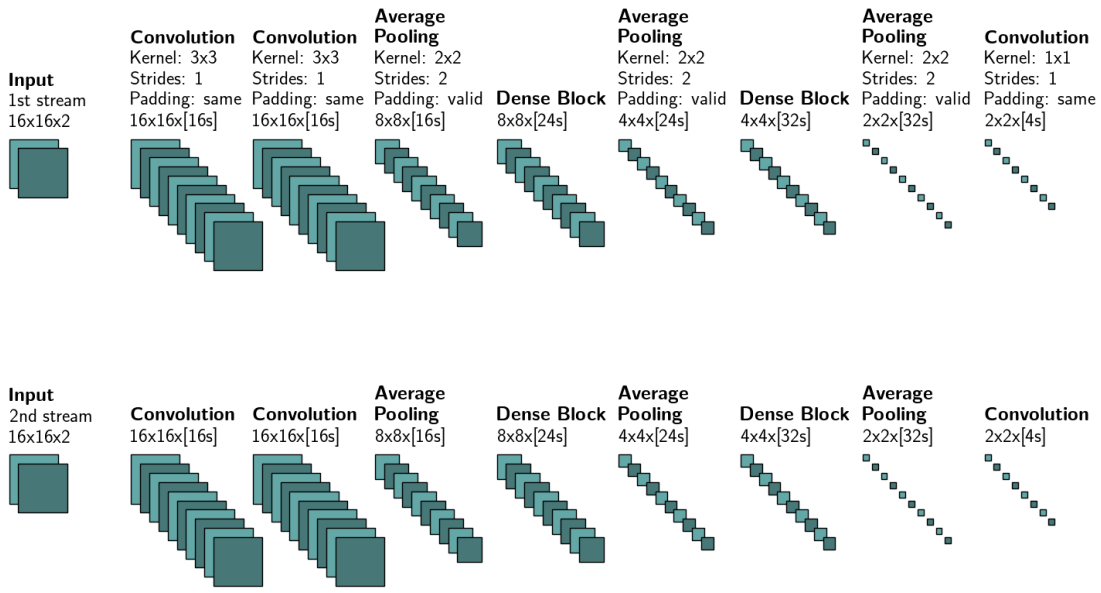


Figure A.5: Architecture of the siamese network with parameter $s \in \mathbb{R}_{>0}$. For example, setting $s = 1.5$ yields $[16s] = [16 \cdot 1.5] = 36$.

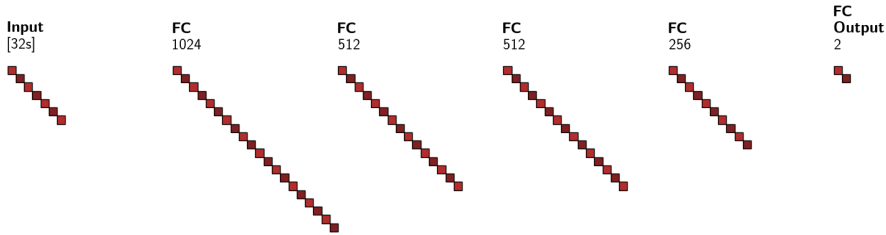


Figure A.6: Architecture of the fully connected network with parameter $s \in \mathbb{R}_{>0}$.

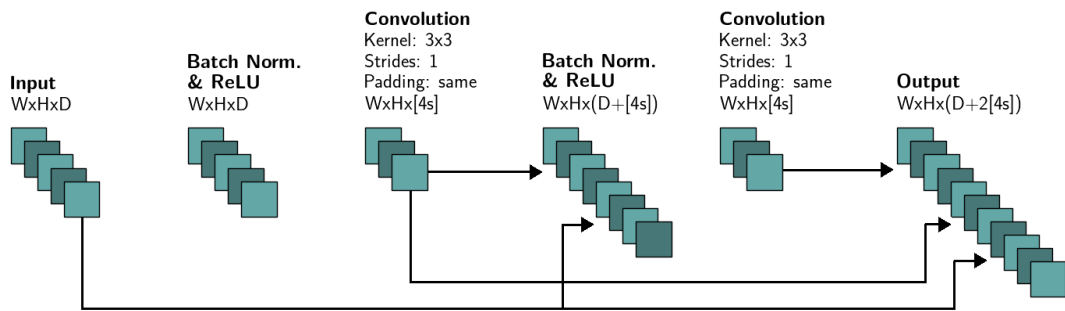


Figure A.7: Architecture of the dense blocks with parameter $s \in \mathbb{R}_{>0}$. The input has size $(W, H, D) \in \mathbb{N}^3$.

Table A.2: Accuracy of the network, rounded to one decimal place, on the test set and the additional point clouds for different network sizes and epochs.

		epochs	
		5	10
$s = 1.0$	test data set	82.5%	89.8%
	additional point cloud	51.0%	50.4%
$s = 1.5$	test data set	94.5%	99.8%
	additional point cloud	52.8%	51.3%
$s = 2.0$	test data set	99.4%	100%
	additional point cloud	50.9%	52.8%

transformed using a batch-normalization, ReLU, and a convolution layer with a fixed output depth of $4s$ channels. Then, the output of the convolution layer and the input of the dense block are concatenated and transformed using the same sequence of batch-normalization, ReLU and convolution layers. This results in the output of the dense block as the concatenation of the input, the first transformation and the second transformation.

Figure A.6 depicts the metric learning network. It consists of five fully connected (fc) layers with ReLU activation functions, except for the last layer, which applies a softmax activation function. As input, the metric learning network obtains the the flattened concatenation of the output of the bottlenecks of both streams. As output, it exports the probability of both descriptions being a matching pair.

The training process is performed to minimize the binary cross-entropy loss added with a l_2 -regularization term on the network parameters to counteract the overfitting. The total data set consists of nine different point clouds sampled with on-board LiDAR sensors with 544600 voxel grid pairs of which 292552, i. e. 54%, are matches. As the number of different point clouds is relatively small compared to the number of voxel grid pairs, it is expected that the network achieves a high accuracy on the data used for training and a low accuracy on new data. Therefore, eight point clouds with 401220 voxel grid pairs are used for training the network, where 55% match. This remaining data set is divided into two equally sized sets for training and testing, yielding a number of training voxel grid pairs is 200610. The batch size is set to 32 and the regularization parameter to $\eta = 5 \times 10^{-4}$. For training, Adam optimizer is carried out with a learning rate of 10^{-4} and different network sizes (parameters $s = 1$, $s = 1.5$, and $s = 2$) and number of epochs (5 and 10). Then, the network is retained with a learning rate of 10^{-5} and the same number of epochs and network sizes as in the first training phase. The whole training time on an NVIDIA GeForce 940M graphics card is about one to five hours, depending on the size of the network.

The test and the remaining point cloud are applied to evaluate the matching accuracy of the network. The accuracy for different network sizes and number of epochs is explained in Table A.2. The highest accuracy of 100% is reached with $s = 2$ and 10 epochs. The lowest accuracy of approximately 50% is reached for every parameter on the additional point cloud. This confirms the assumptions.

Further, the true positive rate depending on the false positive rate are examined with the ROC curves. Figure A.8 illustrates the ROC curves for the test data set. They support the results of Table A.2 and it can be seen that the ROC curve for $s = 2$ with 10 epochs is nearly ideal.

The ROC curves for the additional point cloud are visualized in Figure A.9. It can be seen that all trained networks are approximately guessing the label. The best ROC curves are achieved for $s = 1.5$ with 5 epochs as well as for $s = 2$ with 10 epochs.

Summarizing, the results with the test data set show that the siamese network can in principal determine similar descriptions for points lying on objects with similar curvatures. However, this only holds to similar scenes which are contained in the training set. It can be seen that the network is not

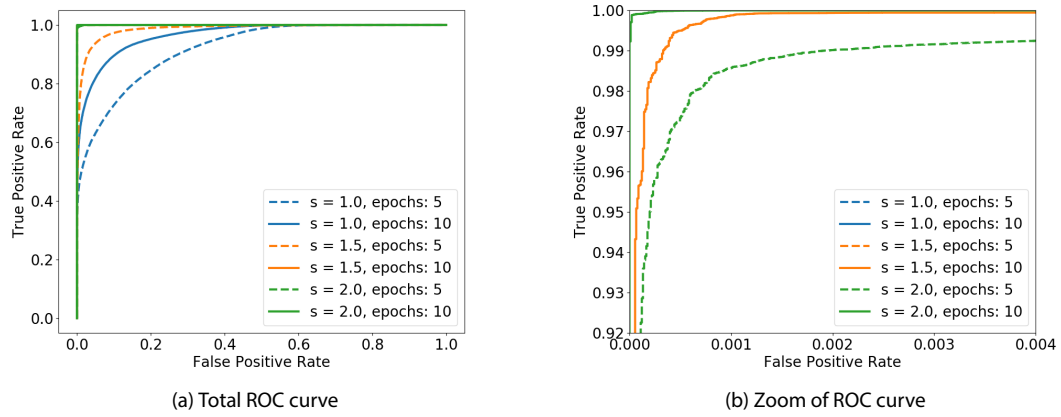


Figure A.8: ROC curves using the test data set for different network sizes and number of epochs

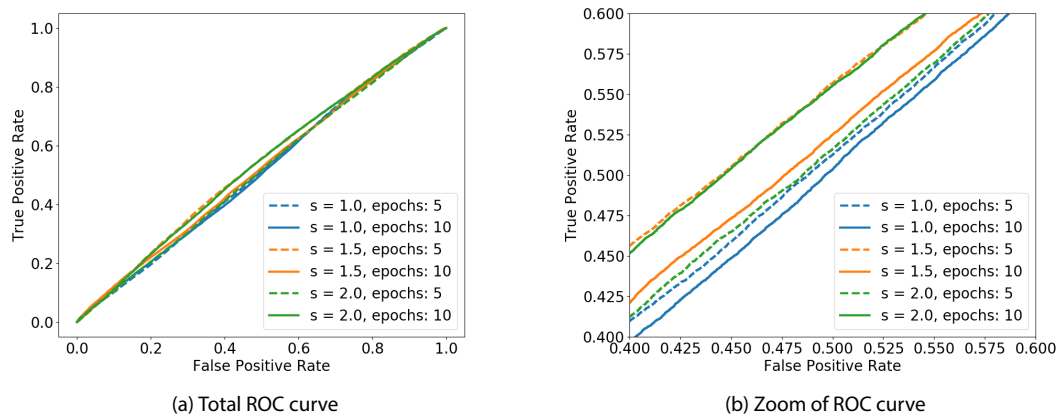


Figure A.9: ROC curves using the additional point cloud for different network sizes and number of epochs

Table A.3: Validation accuracies of the description clustering method for autoencoder one for different regularization parameters. The best accuracy (acc) for each penalty weight α is reported with the epoch (ep) in which it was reached. For an explanation of the abbreviations sdds and ssdd we refer the reader to Table A.1. True and False describe if a bias was used or not. All accuracies of 100% are marked green, while red markings depict runs where the output of the autoencoder is random.

α	sdds2				sdds4				ssdd2				ssdd4			
	True		False		True		False		True		False		True		False	
	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep
0.000	0.855	21	0.998	1	0.780	15	0.999	12	0.879	16	0.994	1	0.988	9	0.968	13
0.012	0.999	2	0.999	20	0.992	19	0.996	22	0.764	1	0.878	5	0.996	5	0.992	17
0.024	1.000	13	0.996	16	0.999	1	0.999	19	0.989	4	0.981	22	0.995	5	0.995	1
0.036	0.999	1	0.996	10	0.996	3	0.996	15	0.981	18	0.995	1	0.997	2	0.992	2
0.048	0.500	1	0.994	2	0.959	9	0.999	7	0.932	5	0.995	3	0.500	2	0.990	10
0.060	0.998	2	0.999	1	0.996	4	1.000	1	0.999	8	0.946	1	0.994	8	0.991	15
0.072	0.999	2	0.999	5	0.995	15	1.000	3	0.981	21	0.941	7	0.999	7	0.990	14
0.084	0.999	5	1.000	5	0.998	1	0.999	11	1.000	4	0.739	7	0.500	1	0.994	2
0.100	0.996	1	0.998	21	0.992	1	0.999	6	0.980	12	0.977	11	0.500	2	0.993	3
0.120	1.000	16	0.990	3	0.996	1	0.997	3	1.000	6	1.000	32	1.000	3	0.999	26
0.140	1.000	1	0.993	7	0.998	4	0.996	13	0.997	20	0.991	18	0.833	28	1.000	17
0.160	0.501	1	1.000	1	0.999	1	1.000	1	1.000	2	1.000	29	1.000	7	0.998	12
0.180	0.500	2	1.000	9	0.997	13	1.000	3	1.000	16	0.999	5	1.000	6	0.999	37
0.200	0.998	11	0.999	3	0.999	1	0.999	14	1.000	3	0.999	10	1.000	11	1.000	5
0.220	0.500	1	1.000	5	0.998	1	1.000	3	0.968	24	1.000	20	0.529	1	1.000	14
0.240	0.996	7	1.000	1	0.999	1	0.999	9	1.000	11	0.993	1	0.502	6	0.995	37
0.260	0.994	10	0.999	3	0.989	18	0.998	11	0.882	20	0.999	40	0.525	1	1.000	25
0.280	0.993	16	0.999	18	0.500	1	0.999	2	0.882	7	0.999	32	0.960	31	1.000	14
0.300	0.500	3	0.989	10	0.500	1	1.000	1	0.764	1	0.999	19	0.501	7	1.000	4
0.320	0.500	1	1.000	2	0.500	1	1.000	3	0.882	7	0.999	25	1.000	6	0.992	39
0.340	0.992	1	1.000	1	0.998	5	0.997	9	0.764	1	0.820	17	0.500	1	1.000	19
0.360	0.500	1	1.000	2	0.995	21	0.998	5	0.764	1	1.000	11	0.507	3	0.999	14
0.380	0.500	2	0.999	15	0.500	1	0.996	8	1.000	10	0.969	3	1.000	3	0.999	5
0.400	0.500	1	1.000	2	0.500	1	1.000	2	0.882	3	0.997	24	0.500	6	0.999	26
0.420	0.500	1	0.998	3	0.500	1	0.999	4	0.956	23	0.997	20	0.998	32	0.999	11
0.440	0.500	2	0.994	2	0.500	1	0.999	1	1.000	8	0.998	32	1.000	5	0.981	36
0.460	0.998	1	1.000	1	0.500	1	0.999	10	1.000	6	0.999	39	0.500	3	0.999	39
0.480	0.500	1	0.999	12	0.998	1	0.994	5	0.882	8	1.000	3	0.501	21	1.000	14
0.500			0.998	1	0.997	18	0.999	12	1.000	4	1.000	27	1.000	4	0.998	34

able to generalize, when testing on a completely new scene. This coincides with the expected results mentioned before. In order to overcome this disadvantages, it is recommended to apply a training set which contains many different point clouds and only a small number of voxel grids of each point cloud. This would avoid overfitting of the network to the point clouds used in the training set.

Table A.4: Validation accuracies of the description clustering method for autoencoder two for different regularization parameters. The best accuracy (acc) for each penalty weight α is reported with the epoch (ep) in which it was reached. For an explanation of the abbreviations sdds and ssdd we refer the reader to Table A.1. True and False describe if a bias was used or not. All accuracies of 100% are marked green, while red markings depict runs where the output of the autoencoder is random.

α	sdds2				sdds4				ssdd2				ssdd4			
	True		False		True		False		True		False		True		False	
	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep	acc	ep
0.000	0.993	8	0.998	1	1.000	1	0.993	1	0.981	21	0.987	1	0.996	22	1.000	3
0.012	0.996	14	1.000	3	0.996	4	0.999	2	0.764	1	0.998	8	0.998	4	0.998	22
0.024	1.000	9	0.993	3	0.997	4	0.990	12	0.992	17	0.995	17	0.999	2	0.997	1
0.036	1.000	1	0.992	6	1.000	1	0.987	15	0.998	1	0.981	7	0.995	21	0.998	2
0.048	0.500	1	1.000	1	0.999	1	1.000	2	0.985	6	0.999	1	0.997	1	0.999	2
0.060	0.999	1	1.000	2	0.999	1	1.000	1	0.982	2	1.000	3	0.997	1	1.000	13
0.072	0.999	1	1.000	1	0.990	13	1.000	4	0.994	17	1.000	1	0.996	2	1.000	7
0.084	0.997	5	0.998	5	0.999	1	1.000	3	1.000	1	0.997	9	0.500	1	0.999	5
0.100	0.995	5	1.000	7	0.992	16	0.995	1	0.999	12	0.997	4	0.997	3	1.000	18
0.120	1.000	9	0.976	1	0.995	1	0.994	1	1.000	9	1.000	37	1.000	3	1.000	1
0.140	1.000	1	1.000	1	1.000	14	0.993	4	1.000	9	1.000	2	0.920	16	0.997	29
0.160	0.500	2	0.994	6	0.946	15	0.995	8	1.000	3	1.000	7	1.000	6	0.961	1
0.180	0.500	2	1.000	1	0.999	6	0.864	19	1.000	15	1.000	2	1.000	6	0.992	28
0.200	0.997	5	1.000	2	0.999	1	0.995	3	1.000	3	0.999	3	1.000	12	1.000	5
0.220	0.500	1	1.000	1	0.501	4	1.000	3	0.952	23	1.000	1	0.508	1	1.000	1
0.240	0.501	7	0.985	1	1.000	1	1.000	1	0.999	11	1.000	1	0.505	1	1.000	1
0.260	0.992	3	1.000	7	0.501	21	1.000	8	0.882	10	1.000	3	0.501	11	1.000	9
0.280	0.500	6	1.000	3	0.500	1	1.000	1	0.882	10	1.000	13	0.501	19	1.000	3
0.300	0.500	3	0.988	18	0.500	1	1.000	2	0.882	5	1.000	1	0.501	5	1.000	1
0.320	0.500	1	1.000	1	0.500	3	1.000	5	0.764	2	1.000	1	1.000	6	0.997	1
0.340	0.995	1	0.980	1	0.998	5	1.000	2	0.764	1	0.999	1	0.501	8	1.000	19
0.360	0.500	1	1.000	1	0.990	7	1.000	1	0.764	1	1.000	1	0.921	1	0.993	2
0.380	0.500	4	1.000	4	0.500	1	0.992	7	1.000	23	1.000	9	1.000	3	0.994	4
0.400	0.500	1	1.000	2	0.500	2	1.000	2	0.764	1	1.000	26	0.501	34	1.000	33
0.420	0.501	2	0.997	1	0.500	1	1.000	2	0.975	12	1.000	19	0.989	22	1.000	1
0.440	0.999	1	0.996	5	0.500	1	1.000	1	1.000	12	0.997	22	1.000	30	0.998	31
0.460	0.993	20	0.997	1	0.500	1	0.993	14	1.000	5	0.997	1	0.501	11	1.000	3
0.480	0.500	1	0.995	15	0.998	1	0.994	8	0.882	3	1.000	2	0.502	7	1.000	1
0.500			0.998	6	0.998	20	1.000	1	1.000	3	1.000	1	1.000	40	0.988	39

B. Number of Clusters in Description Space

For the detection of Feature Areas (FAs), a two-staged clustering process is performed, as explained in Section 4.2. A FA is a set of connected points of a LiDAR point cloud with similar geometry-based descriptions. The FAs generation is based on the main idea of a two-stepped clustering approach, considering both the point cloud and their descriptions.

Initially, descriptions of points with enough points in their local neighborhood are determined with one descriptor. Following, the two-stage clustering method is performed. For this reason, the first clustering step groups points with similar descriptions. These groups are individually used to cluster points into areas including spatially connected points, only considering suited point of the point cloud. In this way, multiple spatially connected areas, i. e. non-semantic objects, with similar descriptions are obtained, i. e. the FAs.

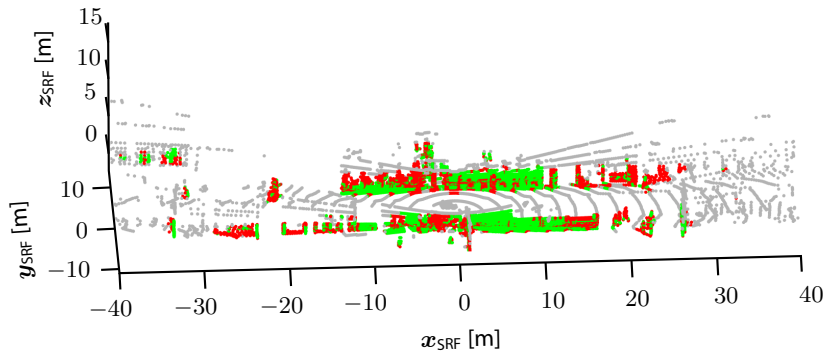
In the following sections, details about the clustering in the description space and the selection of suited points for the Euclidean clustering are depicted.

In this section, more details about the first clustering step, the clustering in the description space, are explained. As the k -medoids method is selected as clustering algorithm a number of clusters must be defined previously, cf. Subsection 2.2.7. In Subsection 4.4.1 it is argued that two clusters are appropriate in this step to avoid over- and underclassification.

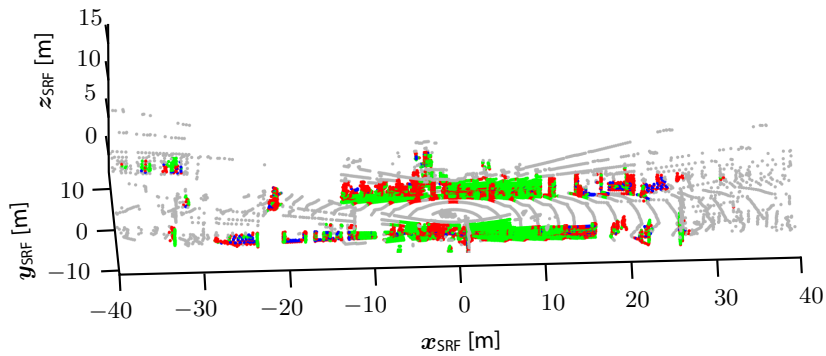
Here, it is explained why a grouping into two clusters is reasonable. Therefore, Figure B.1 shows the results of the description clustering with the k -medoids into two, three, or four clusters. It is illustrated based on a sample point cloud recorded with a Velodyne VLP 32-C LiDAR sensor in a typical suburb area. The descriptions are calculated with the FPFH descriptor. Points as part of the same cluster are colored either in red, green, blue, or yellow. The points marked in gray are rejected in the preprocessing step of the FA extraction.

It can be seen that a typical environment does not have a large variety of different geometries of objects. This is largely caused by the poor sampling of on-board sensors. They prevent the capturing of details, which would enhance the descriptions' variety. Therefore, too many classes causes overclassification. This is obvious when clustering into four classes, see B.1c. Walls are divided into multiple parts, even when the sampling rate is relatively large. Especially at larger distances, where the sampling rate becomes smaller, the clustering into three or four groups leads to noisy clusters.

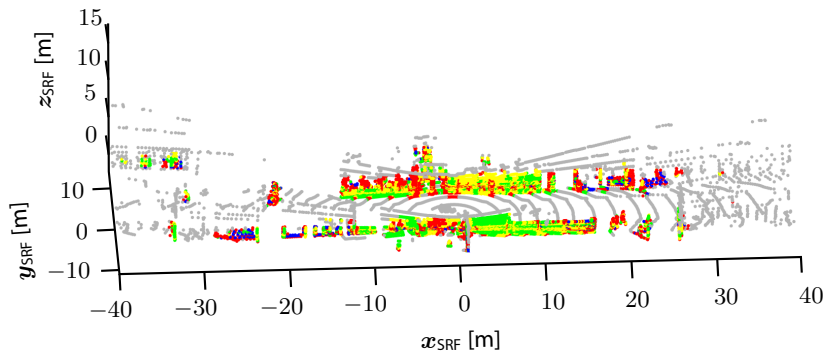
Therefore, clustering into more than two groups is not advisable when dealing with real-world data. Otherwise, the clustering would include a higher computational effort and yield less practical results.



(a) Two classes



(b) Three classes



(c) Four classes

Figure B.1: Clustering of a sample point cloud recorded with a Velodyne VLP 32-C in the description space with k -medoids into two, three, or four clusters through the FPFH descriptor. Each point belonging to the same cluster is colored either in red, green, blue, or yellow. Gray points are sorted out in the FA extraction's preprocessing step.

C. Feature Area Extraction with Intensity-based Descriptors

In the first step of the *LiDAR-Feature-based Localization*, descriptions for every point of the point cloud are determined. This results in data which requires a lot of memory. In addition, not every point contributes to the localization. Therefore, the second step of the *LiDAR-Feature-based Localization* extracts FAs automatically which provide useful information for the localization. As Chapter 4 concentrates on the extraction with geometry-based descriptors, this chapter provides a procedure for intensity-based descriptors on the example of the descriptor developed in this thesis, called GRAdients of Intensities as a Local descriptor (GRAIL), refer to Subsection 3.3.2. The intensity-based extraction is also introduced in [52].

The extraction is performed in the same way for both on-board and map point clouds. Initially, for every point of the point cloud whose local neighborhood is large enough, the intensity-based descriptions are determined, refer to Definition 3.5. Secondly, the extraction is performed to select those points whose descriptions are distinctive, cf. Definition 4.14. Concluding, the LIG to extract FAs of intensity-based descriptors, in this case the GRAIL descriptor, only consists of the distinctiveness for on-board as well as for map point clouds. It is called intensity LIG. As intensity data only provides one-dimensional information, the diversity of descriptions of a point cloud is low. Thus, the number of robust elements in the environment described with intensity-based descriptor algorithms is small. Hence, the spatial diversity and the uniqueness is discarded for the intensity LIG only considering the distinctiveness.

Compared to geometry-based descriptor algorithms, the intensity LIG and the intensity distinctiveness are defined for three-dimensional local neighborhoods, not two-dimensional Feature Area Representatives (FARs).

Definition C.1 (Intensity Distinctiveness). For each intensity-based descriptor D , $LIG_{d,D}^i$ is a function

$$LIG_{d,D}^i : \mathcal{P}(\mathbb{R}^3 \times [0, 1]) \times \mathbb{R}^m \rightarrow [0, 1].$$

It assigns a value to a large enough local neighborhood which captures how rare and distinguishable the description of a large enough local neighborhood is.

Definition C.2 (Intensity Localization Information Gain). Let $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}) \in \mathcal{P}(\mathbb{R}^3 \times [0, 1]) \times \mathbb{R}^m$ be a large enough local neighborhood with radius $r \in \mathbb{R}_{>0}$ around the j -th point $\mathbf{p}_j^{x,y,z} \in \mathbb{R}^3$ of a point cloud \mathcal{P} with some metric m , let $\mathbf{d} \in \mathbb{R}^m$ be its intensity-based description, and let $LIG_{d,D}^i(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}))$ be the intensity distinctiveness of $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$. Then, the intensity Localization Information Gain of a large enough local neighborhood $\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z})$ consists of one component

$$LIG^i : \mathcal{P}(\mathbb{R}^3 \times [0, 1]) \times \mathbb{R}^m \rightarrow [0, 1]$$

$$\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}) \mapsto [LIG_{d,D}^i(\mathcal{N}_{\mathcal{P},r}(\mathbf{p}_j^{x,y,z}))]$$

stating the benefit for localization of each computed large enough local neighborhood determined with intensity-based descriptor algorithms.

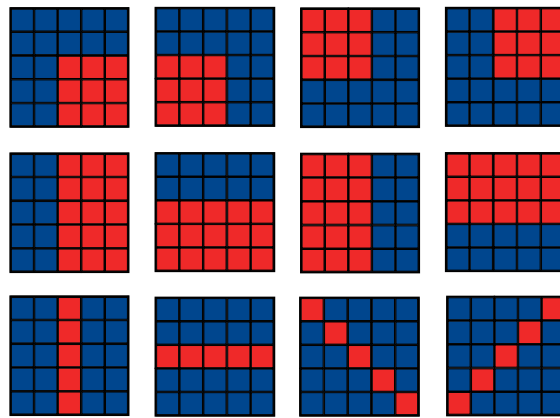


Figure C.1: Distinctive shapes of the intensity patterns the GRAIL descriptor can differentiate between. Pattern cells with maximum reflectivity values are colored red. Pattern cells with zero intensity or no points falling into that cell are colored blue [52].

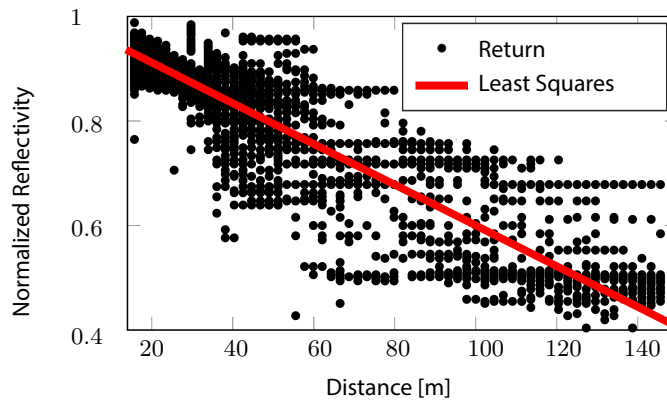


Figure C.2: The functional relation between the normalized reflectivity returns of a Velodyne VLP 32-C and the distance to a planar, highly reflective material is approximated by a least square regression [52].

In case of the GRAIL descriptor, the distinctiveness is measured binarily using the shapes of the 2D grid and the amount of the intensity return. Only descriptions which resulted from distinctive shapes, which often occur in the environment, are considered as distinctive. For that, real-world data was examined to select shapes which can be found frequently, refer to Subsection 3.1.2 for the applied data set. The investigations show that twelve different shapes are suitable for the extraction, which can be seen in Figure C.1. This has three reasons. First, the descriptor can distinguish only between a number of distinctive forms, since the gradients of the intensity image are calculated for eight directions only. Therefore, the number of shapes to be considered are limited. Here, linear combinations or minor deviations of these shapes are not taken into account. Second, the shapes must be dissimilar to one another in order to be able to identify them with certainty. To ensure that this is true for the twelve shapes the distance metric is determined with Equation 3.21 and can be seen in Table C.1. Third, minor changes of the shapes result in minor changes of the description values. Table C.2 illustrates that this holds for the twelve picked shapes. The distances between the minor changes of the shapes and the original shapes, refer to Figure C.1, are computed with Equation 3.21. Here, only the basic forms are investigated as their rotation does not result in different distances.

Table C.1: Distance metric of twelve distinctive shapes of GRAIL in percent





























												
	0	27.3	27.3	42.16	12.26	36.02	36.02	12.26	21.92	21.92	35.03	41.53
	-	0	42.16	27.3	12.26	36.02	12.26	36.02	21.92	21.92	41.53	35.03
	-	-	0	27.3	36.02	12.26	36.02	12.26	21.92	21.92	41.53	35.03
	-	-	-	0	36.02	12.26	12.26	36.02	21.92	21.92	35.03	41.53
	-	-	-	-	0	33.95	26.77	26.77	30.5	14.23	41.74	41.74
	-	-	-	-	-	0	26.77	26.77	30.5	14.23	41.74	41.74
	-	-	-	-	-	-	0	33.95	14.23	30.5	41.74	41.74
	-	-	-	-	-	-	-	0	14.23	30.5	41.74	41.74
	-	-	-	-	-	-	-	-	0	29.45	37.36	37.36
	-	-	-	-	-	-	-	-	-	0	37.36	37.36
	-	-	-	-	-	-	-	-	-	-	0	47.73
	-	-	-	-	-	-	-	-	-	-	-	0

Table C.2: Description differences of minor changes of shapes of GRAIL in percent

	4.23
	5.24
	2.72
	13.09

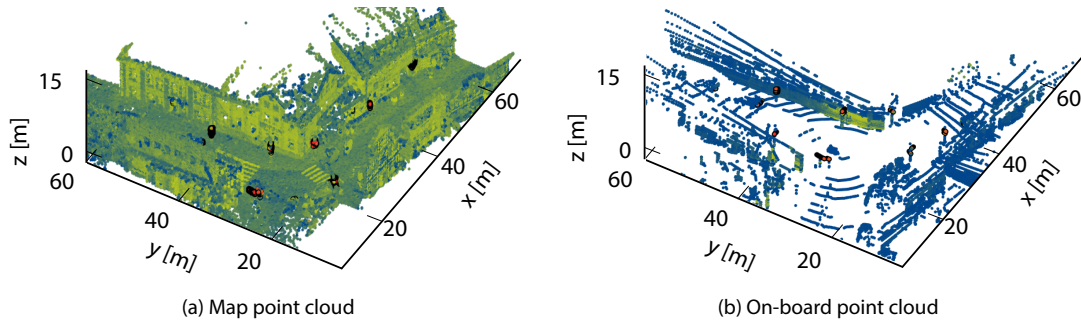


Figure C.3: Sample key points extracted with selection method for the GRAIL descriptor in a dense map and sparse on-board point cloud in a suburb area. The color visualizes the intensity values (red = high, blue = low). The extracted key points are circled in black [52].

Additionally, those points are extracted whose maximal intensity within its 2D grid is greater than a predefined threshold. As the software of many sensors, like the Velodyne VLP-32C, internally compensate for the squared energy loss by the traveled distance of the light pulses this is considered, here. However, Figure C.2 visualizes that the values of the sensors' intensity returns still decrease linearly depending on the traveled distance, here, in case of the Velodyne VLP-32C sensor. Hence, in this thesis, the characteristic curve of the intensity loss for a Velodyne VLP-32C is determined, see Figure C.2. This is done by driving towards a highly reflective material to establish the reflectivity as a function r of the measured distance d . The curve is approximated with a least squares regression to yield the functional relation:

$$\begin{aligned} r : \mathbb{R} &\rightarrow [0, 1] \\ d &\mapsto -0.004d + 0.99. \end{aligned} \quad (\text{C.1})$$

Therefore, in this thesis, the maximal intensity threshold t is distance dependent resulting in:

$$\begin{aligned} t : \mathbb{R} &\rightarrow [0, 1] \\ d &\mapsto -0.004d + 0.99 - 0.2, \end{aligned} \quad (\text{C.2})$$

i. e. 0.2 smaller than the maximal intensity measured at each distance. This threshold was determined in practical experiments, refer to Subsection 3.1.2 for the data set.

Thus, those points are considered as distinctive whose descriptions do not differ more than 15% from that of the twelve shapes of Figure C.1 and whose intensity is large enough considering t . In real-world data environment, this threshold has been shown as feasible, refer to Subsection 3.1.2 for the data set, and is often a lot smaller than the differences between the shapes itself, refer to Table C.1, except for the intensity edge, which is very similar to the corner. Then, the extracted points of the point cloud are clustered using the DBSCAN by Ester et al. [29], see Subsection 2.2.6, to yield a set of connected points.

?? depicts a sample point cloud used for the map generation of a suburb area and an on-board point cloud together with their key points extracted with this method.

