Multi-Modal Continuous-Time SLAM and Semantic Parametric Mapping for Autonomous Driving

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

angenommene

Dissertation

von

M.Sc. Haohao Hu

aus Hunan Provinz, China

Tag der mündlichen Prüfung:	19.03.2026
Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr.-Ing. Klaus Dietmayer

Abstract

For autonomous vehicles, understanding the operational environment is crucial. However, due to limitations of current sensor technologies, perception remains challenging - individual sensors cannot fully compensate for occlusions, adverse weather conditions, or limited sensing range. High Definition (HD) maps expand the range of sensor perception, improving safety and efficiency in navigation and decision-making.

Conventional Simultaneous Localization and Mapping (SLAM) systems typically rely on the fusion of measurements through a pose graph trajectory representation, which is inherently complex due to temporal discontinuities. To mitigate this issue, a continuous-time trajectory representation is necessary. Furthermore, to enhance the informativeness of semantic HD maps and reduce map storage demands, long-term stable and driving-relevant features must be parametrically modeled and mapped.

This thesis introduces a continuous-time trajectory SLAM based on uniform B-splines, which integrates measurements with varying timestamps and frequencies. High-frequency motion can be accurately represented, and velocity and acceleration at any given timestamp can be evaluated. Semantic primitive features are parametrically modeled and, during mapping, detected, associated, and optimized in a robust and reliable manner.

The proposed approach is evaluated using a real-world dataset recorded in urban environments. The results demonstrate that continuous-time SLAM, utilizing LiDAR and camera data, outperforms state-of-the-art pose graph SLAM in terms of accuracy, robustness, and efficiency. Additionally, the generated semantic HD maps exhibit high mapping accuracy and consistency with sensor

data. The automatic mapping framework significantly reduces mapping costs and facilitates scalable map-building processes.

Use of Large Language Models: Large language models (LLMs), including ChatGPT and GitHub Copilot, were used to support the linguistic and grammatical refinements of the text in this thesis. They were used to improve wording and textual structure but not to create new scientific content. All revised text was manually reviewed and further edited by the author.

Kurzfassung

Für automatisierte Fahrzeuge ist eine verlässliche Umgebungswahrnehmung entscheidend. Aufgrund der Begrenzungen aktueller Sensortechnologien bleibt sie jedoch herausfordernd - einzelne Sensoren können Sichtverdeckungen, schlechte Wetterbedingungen oder eingeschränkte Reichweiten nicht vollständig kompensieren. High Definition (HD)-Karten erweitern die Sensorwahrnehmung und erhöhen Sicherheit und Effizienz bei Navigation und Verhaltensgenerierung.

Aktuelle Simultaneous Localization and Mapping (SLAM)-Systeme fusionieren Messungen über einen Posengraphen, der aufgrund zeitlicher Diskontinuitäten inhärent komplex ist. Eine zeitlich kontinuierliche Trajektoriendarstellung adressiert dieses Problem. Zugleich sollten langfristig stabile, fahrrelevante Merkmale parametrisch modelliert und kartiert werden, um semantische HD-Karten informativer zu gestalten und auch den Speicherbedarf zu senken.

Diese Dissertation stellt ein zeitlich kontinuierliches Trajektorien-SLAM auf Basis uniformer B-Splines vor, das Messungen mit unterschiedlichen Zeitstempeln und Frequenzen integriert. Damit lassen sich hochfrequente Bewegungen präzise abbilden, sowie Pose, Geschwindigkeit und Beschleunigung zu beliebigen Zeitpunkten bestimmen. Darauf aufbauend werden semantische Primitive parametrisch modelliert und während der Kartierung zuverlässig detektiert, assoziiert und robust optimiert.

Die Evaluierung auf einem realen städtischen Datensatz zeigt, dass zeitlich kontinuierliches SLAM mit LiDAR- und Kameradaten in Genauigkeit, Robustheit und Effizienz Graph-SLAM übertrifft. Die erzeugten semantischen HD-Karten

sind geometrisch und semantisch konsistent mit den Sensordaten. Das automatische Kartierungssystem senkt die Kartierungskosten deutlich und ermöglicht skalierbaren Kartenaufbau.

Acknowledgements

Die vorliegende Dissertation wurde während meiner Tätigkeit am Institut für Mess- und Regelungstechnik (MRT) des Karlsruher Instituts für Technologie (KIT) verfasst.

Mein besonderer Dank gilt Prof. Dr.-Ing. Christoph Stiller für die Betreuung dieser Arbeit sowie für die Möglichkeit, in dem inspirierenden Forschungsumfeld des MRT eigenständig wissenschaftlich tätig zu sein. Ebenso spreche ich Prof. Dr.-Ing. Klaus Dietmayer meinen herzlichen Dank für die Übernahme des Korreferats und seine wertvollen Anmerkungen aus.

Diese Arbeit wäre ohne die Unterstützung vieler Menschen nicht realisierbar gewesen. Mein besonderer Dank gilt zunächst Jannik Quehl, der meine Masterarbeit betreut und mich ermutigt hat, meine Promotion am MRT fortzusetzen.

Den aktuellen und ehemaligen Kolleginnen und Kollegen des MRT danke ich für die herausragende Zusammenarbeit, die weit über das Berufliche hinausging. Insbesondere möchte ich Jan-Hendrik Pauls, Martin Lauer, Marc Sons, Johannes Beck, Sascha Wirges, Frank Bieder sowie Fabian Immel, Nils Rack, Richard Schwarzkopf, Alexander Blumberg, Jonas Merkert und Johannes Fischer für die zahlreichen anregenden Diskussionen, die konstruktive Kritik und das wertvolle Feedback zu dieser Arbeit danken.

Mein Dank gilt ebenso den nicht-wissenschaftlichen Mitarbeiterinnen und Mitarbeitern, insbesondere Erna Nagler, Alexandra Stotz und Werner Paal, deren tatkräftige Unterstützung und Engagement maßgeblich zum reibungslosen Ablauf am Institut beigetragen haben.

Von Herzen danke ich meinen Eltern, die mich stets in meiner akademischen Laufbahn gefördert und unterstützt haben. Mein tief empfundener Dank gilt

schließlich meiner Frau Na Shen und meiner Tochter Shenxi (Emily) Hu – eure Unterstützung, Geduld und euer Rückhalt waren für mich eine unverzichtbare Kraftquelle während der gesamten Promotionszeit.

Karlsruhe, im März 2026 *Haohao Hu*

Contents

Appendix

Notation

This chapter introduces the notation and symbols which are used in this thesis. In cases where a symbol has more than one meaning, the context (or a specific statement) resolves the ambiguity.

Mathematical Foundations

$\mathbb{R}^+$	Positive real numbers
$\mathbb{R}^2$	2D real space
$\mathbb{R}^3$	3D real space
$\mathbb{R}^n$	nD real space
$\mathbb{R}^{2\times2}$	2x2 real matrices
$\mathbb{R}^{3\times3}$	3x3 real matrices
SE(3)	Special Euclidean group in 3D
SO(3)	Special Orthogonal group in 3D
so(3)	Lie algebra of SO(3)
$\mathbf{I}_\mathrm{d}$	Identity matrix
$\ominus$	Ominus operator
μ	Gaussian mean
Σ	Gaussian covariance
Ω	Information matrix

Spatial Representations and Transformations

$\mathcal{F}_{\mathrm{A}}$	Coordinate system A
$\mathbf{R}$	Rotation matrix
$\mathbf{t}$	Translation vector
$\mathbf{T}$	Isometric transformation / pose
$\mathbf{T}_{\mathrm{A}}^{\mathrm{B}}$	Pose from coordinate system $\mathcal{F}_{\mathrm{A}}$ to $\mathcal{F}_{\mathrm{B}}$
$\mathbf{q}$	Quaternion
φ	Axis-angle vector
$\mathbf{e}$	Axis-angle unit rotation axis
ψ	Axis-angle rotation angle
$\mathbf{X}$	Bézier curve
$\mathbf{S}$	(Uniform) B-spline
$\mathbf{S_t}$	Translation uniform B-spline
$\mathbf{S_R}$	Rotation uniform B-spline
B	Bernstein polynomial

Vehicle Dynamics and Motion

ϕ	Azimuth angle
θ	Elevation/ Polar angle
$\mathbf{v}$	Velocity
$\mathbf{a}$	Acceleration
ω	Angular velocity
$\dot{\omega}$	Angular acceleration

Sensing and Reconstruction

$\mathbf{I}$	Image
$\mathbf{u}$	Pixel point
$\breve{\mathbf{u}}$	Homogeneous pixel point
$\mathbf{u}_{\mathrm{c}}$	Principal point
f	Focal length
$\mathscr{P}_{\mathbf{F}}$	Camera forward model
$\mathscr{P}_{\mathbf{B}}$	Camera backward model
$\mathbf{K}$	Camera intrinsic matrix
π	Camera projection function
ρ	View ray
ρ_{s}	View ray support point
ρ_{d}	View ray direction
$\mathscr{M}_{\mathrm{dt}}$	Distance transform map
$\mathbf{p}$	Point in 2D/ 3D space
$\mathbf{p}_{\mathrm{A}}$	Point in coordinate system $\mathscr{F}_{\mathrm{A}}$
$\mathbf{n}$	Normal vector
$\mathscr{V}$	Voxel
$\mathbf{l}$	Landmark
$\mathscr{L}$	Set of landmarks
$\mathbf{z}$	Feature point
$\mathscr{Z}$	Set of feature points

Optimization and Estimation

$\mathbf{r}$	Residual vector
$\mathbf{x}$	Parameter vector
$\hat{\mathbf{x}}$	Optimal parameter vector
$\tilde{\mathbf{x}}$	Measured parameter vector
$\mathbf{J}$	Jacobian matrix

1 Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental technology that enables autonomous systems to simultaneously construct a map of their environment and estimate their position within it. It plays a crucial role in various domains, including robotics, autonomous vehicles, and Augmented Reality (AR). For autonomous driving, SLAM facilitates the construction of environmental maps, enabling robust navigation. The decision-making process of autonomous vehicles relies on environmental perception, which is inherently constrained by the physical properties of sensors. To enhance the reliability of perception, SLAM techniques aggregate sensor measurements over time to build a consistent environmental map, compensating for the limited instantaneous sensing range.

The most commonly used sensors in SLAM approaches for autonomous vehicles include cameras, LiDAR, Inertial Measurement Unit (IMU), and Global Navigation Satellite System (GNSS). Cameras are relatively cost-effective and provide color, texture, and shape information about the environment. LiDAR generates precise and dense 3D point clouds without requiring an external light source. While cameras and LiDAR have functional limitations, an IMU offers valuable motion information. For global localization, GNSS can be used. To effectively integrate measurements from these diverse sensors, robust and accurate fusion algorithms are essential.

An High Definition (HD) map, along with precise localization within it, is essential for autonomous driving. By incorporating HD maps, perception capabilities can be extended, mitigating occlusions and enhancing driving performance. To improve the robustness and reliability of HD maps against changes while enriching their informativeness, long-term stable features should be mapped with semantic labels. Given storage constraints, these features must

be represented in a compact manner. Thus, a semantic mapping framework is necessary to build HD maps with long-term stable and driving-relevant semantic features.

1.1 Problem Statement

Building HD maps for autonomous vehicles requires an accurate and robust SLAM system. To achieve optimal SLAM performance, fusing measurements from multiple sensors, including cameras, LiDAR, IMU, and GNSS, is a logical approach, as they provide complementary information due to their distinct physical properties. However, integrating these diverse measurements is challenging since they represent different physical quantities and typically operate on different scales. Bundle adjustment algorithms with pose graph backends are commonly employed to unify these measurements.

However, a pose graph is a discrete representation of continuous-time motion, consisting of a sequence of poses sampled at a rate constrained by computational cost. Accurately estimating motion between two discrete poses is difficult, even though IMU provides high-frequency measurements. To incorporate IMU data, integration is required, which introduces accumulated errors. Additionally, pose graphs require time-synchronized measurements, which are not always feasible. Sensor measurements often arrive with different timestamps and at varying frequencies. Although interpolation between poses can be used to address this issue, it also introduces errors. Therefore, a continuous-time motion representation should be incorporated into bundle adjustment algorithms to accommodate measurements with different timestamps and frequencies while accurately representing high-frequency motion.

A 3D feature-point map is a key output of SLAM systems and is used for autonomous vehicle localization. However, these point features are low-level, lack semantic information, and are not stable over the long term. Furthermore, feature-point maps require substantial storage, complicating map management. To overcome these limitations, semantic HD maps incorporating long-term

stable features are needed. Such semantic HD maps typically consist of vectorized semantic features that are manually annotated, a process that is time-consuming, error-prone, and costly. Therefore, an automated, scalable semantic mapping framework with vectorized features is required for autonomous vehicles.

1.2 Contributions

The first major contribution of this thesis is the development of a continuous-time trajectory representation for bundle adjustment algorithms using uniform B-splines, offering a compact parametric representation of poses:

- The continuous-time trajectory representation enables the fusion of sensor measurements (cameras, LiDAR, IMU, GNSS) with varying timestamps and frequencies, allowing for accurate high-frequency motion representation.
- The computational complexity of bundle adjustment algorithms is not significantly affected by the number of estimated motion poses.
- A compact axis-angle representation is selected for 3D rotation parameterization, and measurement models for different sensor modalities are formulated for bundle adjustment algorithms.
- Once the continuous-time trajectory is estimated, pose, velocity, and acceleration can be efficiently evaluated at any given timestamp.

The second major contribution is the development of an optimization-based, automatic semantic parametric mapping framework incorporating long-term stable semantic primitive features:

- Semantic primitive features—including elevated objects (poles, traffic lights, traffic signs) and road markings (dashed lines, solid lines, arrows, curbs)—are automatically and accurately mapped using LiDAR point clouds and camera semantic images.

- These semantic primitives are represented with parametric models. Traffic signs of various shapes are modeled with precision, while road markings are represented using 3D polygons and lines.
- Semantic primitive features are tracked over time, and their model parameters are optimized using semantic masks from multiple frames, enhancing robustness and reliability.

1.3 Outline

Chapter 2 provides an overview of existing research in the field of SLAM technology and semantic primitive mapping for autonomous driving.

Chapter 3 presents the required theoretical foundations. It begins with a description of pinhole and spherical camera models, including forward and backward projection functions. Additionally, 3D rigid-body transformations, uniform B-splines, and optimization techniques are introduced.

Chapter 4 details the continuous-time trajectory representation for bundle adjustment algorithms. Various 3D rotation parameterization methods are discussed, and the compact axis-angle representation is selected. Time derivatives for continuous-time trajectories using the axis-angle representation for 3D rotation are demonstrated. Furthermore, measurement models for different sensor modalities are developed and integrated into bundle adjustment algorithms.

In Chapter 5, the optimization-based semantic parametric mapping approach is introduced. Semantic primitive features are categorized into elevated objects (poles, traffic lights, traffic signs) and road markings (dashed lines, arrows, solid lines, curbs). These features are represented with geometric, parametric models. They are detected, tracked, and optimized using LiDAR and camera measurements.

Chapter 6 presents the evaluation of the proposed continuous-time trajectory SLAM and semantic primitive parametric mapping framework. The evaluation is conducted using a real-world dataset collected in urban environments with an

experimental vehicle. SLAM performance is analyzed under different configurations, comparing results with and without LiDAR measurements and using either a continuous-time trajectory or a discrete pose graph backend. Results are assessed both quantitatively and qualitatively.

Finally, Chapter 7 concludes this thesis by summarizing key findings and providing an outlook on future research directions.

2 Related Work

In this section, state-of-the-art SLAM techniques are first discussed. Based on the primary sensor used, SLAM methods can be broadly categorized into three types: visual SLAM, LiDAR SLAM, and visual-LiDAR SLAM. Subsequently, advancements in semantic mapping are reviewed to provide a comprehensive understanding of the current research landscape.

2.1 Visual SLAM

Visual SLAM primarily utilizes cameras as the main sensing modality. It can be broadly classified into two categories based on the types of features utilized: feature-based visual SLAM and direct visual SLAM. Feature-based approaches rely on point features, whereas direct methods utilize raw pixel intensities.

2.1.1 Feature-based Visual SLAM

The most widely adopted feature-based visual SLAM algorithms include PTAM, introduced by Klein et al. [Kle07], and the ORB-SLAM family developed by Mur-Artal et al. [Mur15], Mur-Artal et al. [Mur17], and Campos et al. [Cam21]. PTAM was one of the first systems to demonstrate real-time feature-based SLAM. By parallelizing tracking and mapping, it significantly improved computational efficiency and robustness.

The ORB-SLAM family consists of three variants: ORB-SLAM Mur-Artal et al. [Mur15], ORB-SLAM2 Mur-Artal et al. [Mur17], and ORB-SLAM3 Campos et al. [Cam21]. ORB-SLAM employed ORB features for robust real-time tracking, mapping, and loop closure detection in large-scale monocular

environments. ORB-SLAM2 extended this to stereo and Red Green Blue - Depth (RGB-D) cameras with parallel bundle adjustment for enhanced accuracy. ORB-SLAM3 introduced multi-map management for switching between maps in large-scale and dynamic environments.

Similarly, the approaches introduced by Lategahn [Lat13] and Sons [Son21] utilized camera-based point features for motion estimation and mapping. These methods relied on viewpoint, scale, and rotation-invariant feature points to enhance robustness in motion estimation and loop closure detection. The system introduced by Sons [Son21] extended visual SLAM to multi-session mapping, allowing new trajectories to be aligned with previous maps. Additionally, it incorporated a surround-view camera system, which facilitated the extraction of a larger number of informative feature points.

Recent advancements integrated deep learning techniques for feature extraction, matching, and direct pose estimation. For instance, DeTone et al. [DeT17] introduced the SuperPoint network for simultaneous feature extraction and description, while Sarlin et al. [Sar20] developed SuperGlue for feature-point matching. Similarly, D2-Net developed by Dusmanu et al. [Dus19] and the approach introduced by Hu et al. [Hu21] employed deep learning techniques for feature extraction and description. PoseNet [Ken15] and ViPNet [Hu20] further explored deep learning for direct pose estimation.

Generally, feature-based visual SLAM methods predominantly rely on distinct visual features, such as corners, edges, and textures, to establish correspondences between frames and estimate camera motion. However, these approaches face significant challenges in environments that are textureless, homogeneous, or poorly illuminated, where reliable feature detection and matching become increasingly difficult. The inability to extract and associate robust features in such conditions often leads to substantial localization and mapping errors, ultimately compromising the overall stability and accuracy of the system or even causing complete failure.

2.1.2 Direct Visual SLAM

Direct visual SLAM eliminates the reliance on explicit feature extraction and matching by utilizing raw pixel intensities for localization and mapping. These methods optimize camera poses by directly minimizing the photometric error between observed and projected image intensities.

Prominent direct visual SLAM approaches include LSD-SLAM [Eng14], DSO [Eng18], and Stereo DSO [Wan17]. LSD-SLAM, introduced by Engel et al. [Eng14], demonstrated the feasibility of direct SLAM for large-scale applications. It maintained a semi-dense depth map and aligned frames using direct image alignment. DSO [Eng18] further improved performance by selecting sparse pixels with high-intensity gradients, optimizing camera poses via photometric error minimization within bundle adjustment frameworks. This significantly enhanced computational efficiency and accuracy, making DSO suitable for real-time applications. Stereo DSO [Wan17] extended DSO to stereo cameras, improving depth estimation and robustness.

Compared to feature-based methods, direct visual SLAM is more robust in textureless or low-feature environments, as it does not rely on discrete feature extraction. However, it is computationally expensive due to direct optimization of pixel intensities. Additionally, it is sensitive to lighting variations, a common issue in outdoor environments.

In conclusion, both feature-based and direct visual SLAM approaches present distinct trade-offs. Given the structured nature of urban driving environments, where rich and distinctive visual point features are consistently present, feature-based methods provide a favorable balance between computational efficiency and robustness. For this reason, this thesis adopts a feature-based visual SLAM approach as the visual front-end.

2.2 LiDAR SLAM

As discussed in Section 2.1, visual SLAM is highly dependent on lighting conditions and can fail in low-light or nighttime environments. To overcome this

limitation, LiDAR sensors can be utilized as primary sensors for SLAM. LiDAR operates independently of external illumination and provides precise distance measurements to surrounding objects. Based on the type of features used for motion estimation, the most widely adopted LiDAR SLAM approaches can be categorized into three main types: point-based Iterative Closest Point (ICP), Normal Distributions Transform (NDT), and voxel-based methods.

2.2.1 Point-based ICP

ICP is a deterministic, geometry-based algorithm that iteratively refines alignment between a source and a target point cloud. It operates by establishing point-to-point or point-to-plane correspondences based on a coarse transformation guess and by minimizing distances between matched points.

The classical ICP algorithm, introduced by Besl et al. [Bes92], provided a robust method for aligning 3D shapes using point-to-point matching and distance minimization. GICP, developed by Segal et al. [Seg09], improved upon this by combining point-to-point and point-to-plane ICP, leading to a more adaptive framework that better captures local geometric structure and enhances accuracy. GICP minimized point-to-plane distance for planar surfaces and point-to-point distance for non-planar regions.

Beyond standard ICP, feature-based ICP variants have been proposed, which use subsets of distinctive feature points extracted from original point clouds instead of processing full datasets. By leveraging highly informative feature points, computational efficiency is improved, and robustness is enhanced. A well-known example is the LOAM family. LOAM [Zha14] extracted edge and planar features from LiDAR point clouds to facilitate robust scan matching. In its odometry module, LOAM minimized point-to-edge and point-to-plane distances to estimate relative pose. The mapping module then updated a global map using estimated poses and extracted features. LeGO-LOAM [Sha18] further optimized LOAM for large-scale environments, achieving a balance between computational efficiency and mapping accuracy. It introduced a two-stage scan matching process: ground-plane alignment for roll and pitch estimation, followed by full scan matching for translation and yaw computation.

Although ICP methods achieve high accuracy, their performance is highly dependent on the quality of the initial guess. Poor initialization can lead to convergence toward local minima, resulting in inaccurate alignments. Furthermore, standard ICP algorithms are computationally intensive, as they involve iterative closest-point searches and transformation refinements, often requiring optimized implementations for real-time applications.

2.2.2 NDT

NDT is a widely used LiDAR scan registration technique that models point clouds as probabilistic distributions instead of discrete points. LiDAR data is segmented into a voxel grid, with each voxel representing a local Gaussian distribution estimated from point statistics. Registration is performed by maximizing the likelihood between the distributions of source and target point clouds.

NDT, introduced by Biber et al. [Bib03], demonstrated the feasibility of using normal distributions for point cloud registration, exhibiting robustness in noisy and sparse data conditions. D2D-NDT, developed by Stoyanov et al. [Sto12], extended NDT by replacing point-to-distribution matching with distribution-to-distribution matching, further improving robustness in dynamic and cluttered environments.

In general, NDT-based methods provide a robust and computationally efficient alternative to traditional point-based registration techniques. By leveraging probabilistic representation and grid-based processing, they offer enhanced resilience to noise and sparse measurements, making them well-suited for real-time applications in complex environments.

2.2.3 Voxel-based Methods

As noted in Section 2.2.1, ICP methods suffer from high computational costs due to large numbers of points and iterative search for closest points. Voxel-based methods mitigate this problem by employing a voxel grid to downsample input data and efficiently register LiDAR scans. Like NDT, voxel-based approaches divide 3D space into discrete voxels and process points within each

voxel to improve computational efficiency and robustness. Similar to ICP-based techniques, voxel-based methods often focus on extracting specific features that are critical for accurate registration.

A notable voxel-based method is VGICP Koide et al. [Koi21], which extended GICP by incorporating voxelization to eliminate costly nearest-neighbor searches while preserving accuracy. Unlike NDT, VGICP derives voxel distributions by aggregating individual point distributions within each voxel. Building upon this, BALM Liu et al. [Liu21] introduced bundle adjustment to refine local LiDAR voxel features, further improving accuracy and robustness.

By leveraging voxel-based processing, these methods achieve substantial computational efficiency while maintaining robustness. The incorporation of bundle adjustment techniques into voxel-based registration further reduces drift, enhancing long-term mapping accuracy.

Among the reviewed LiDAR SLAM approaches, BALM Liu et al. [Liu21] offers a favorable combination of accuracy, efficiency, and robustness through its voxel-based point cloud feature representation and bundle adjustment optimization. For this reason, this thesis adopts a BALM-based LiDAR front-end, utilizing voxel point cloud features for scan registration within the continuous-time SLAM framework.

2.3 Visual-LiDAR SLAM

As outlined in Section 2.1 and Section 2.2, both visual and LiDAR SLAM approaches have inherent advantages and limitations. Camera images provide rich texture information, which is highly beneficial for feature extraction and matching. However, cameras are sensitive to illumination conditions and can fail in low-light environments. Conversely, LiDAR provides accurate distance measurements and is robust to lighting variations, but lacks detailed texture information. Visual-LiDAR SLAM combines both sensing modalities to leverage their complementary strengths.

LiMO, introduced by Graeter et al. [Gra18], integrated LiDAR depth with visual feature tracking to improve pose estimation accuracy. V-LOAM, developed by Zhang et al. [Zha15], further enhanced visual-LiDAR SLAM by utilizing depth maps for odometry estimation and mapping. V-LOAM was extended to DV-LOAM by Wang et al. [Wan21] and SDV-LOAM by Yuan et al. [Yua23]. Both methods integrated monocular direct visual odometry with LiDAR depth maps for spatial scale estimation and incorporated LiDAR scan-to-map optimization to improve robustness and accuracy in pose estimation.

Visual-LiDAR SLAM represents a significant advancement by effectively mitigating the limitations of single-sensor approaches. The fusion of visual and LiDAR data enhances robustness against noise and outliers, as inconsistencies in one sensor can be compensated by the other. By leveraging these complementary strengths, visual-LiDAR SLAM improves accuracy, reliability, and scalability in complex environments. Given these advantages, this thesis employs a visual-LiDAR SLAM framework. Instead of relying on a monocular camera, a multi-surround-view camera system is utilized to provide additional depth information and enhance perception capabilities.

The aforementioned SLAM approaches share a common limitation in their optimization backend: they rely on a discrete-time pose graph, where sensor measurements are associated with a finite set of keyframe poses. This discrete representation introduces temporal discontinuities, making it difficult to accurately model high-frequency motions and to fuse sensors operating at different frequencies without resorting to interpolation heuristics. Furthermore, continuous quantities such as velocity and acceleration cannot be directly queried at arbitrary time instants within such a framework. To address these shortcomings, this thesis proposes a continuous-time trajectory representation based on uniform B-Splines as the optimization backend. This formulation defines the vehicle trajectory as a smooth, time-continuous function, enabling the direct integration of measurements with heterogeneous timestamps and sampling rates. As a result, pose, velocity, and acceleration can be evaluated at any desired time instant, allowing for more accurate motion modeling and tighter multi-sensor fusion.

2.4 Semantic Primitive Mapping

The motivation for semantic primitive mapping is that semantic primitives are crucial for autonomous driving, as they are driving-relevant, temporally stable, and enhance the robustness and reliability of maps against environmental changes. Depending on their nature, semantic primitives can be categorized into two types: elevated primitives and road-marking primitives. Elevated primitives exist above the ground surface and include poles, traffic lights, and traffic signs. Road-marking primitives are located on the ground surface and comprise dashed lines, solid lines, arrows, and curbs.

2.4.1 Elevated Primitive Mapping

Elevated primitives, such as poles, traffic lights, and traffic signs, are crucial for autonomous driving due to their long-term stability, static nature, and vertical structure, making them less prone to occlusion.

The work presented by Sefati et al. [Sef17] extracted pole-like landmarks from LiDAR point clouds and stereo camera images. These extracted landmarks were subsequently used to improve vehicle localization. Similarly, the approach proposed by Kümmerle et al. [Küm19b] identified basic geometric primitives, such as cylindrical objects and facades, from LiDAR point clouds for vehicle self-localization. The extraction of pole-like primitives followed a method similar to that of Sefati et al. [Sef17], while building facades were approximated using planar surfaces. Both works relied solely on geometric characteristics without incorporating semantic information.

A more advanced approach, introduced by Pauls et al. [Pau21], parametrically modeled and mapped poles, traffic lights, and traffic signs. Poles and traffic lights were represented as cylinders, whereas traffic signs were approximated as rectangular planes. To estimate model parameters, depth information from LiDAR point clouds was combined with image instance bounding boxes generated by SeamSeg [Por19]. Model parameters were estimated on a per-frame basis and tracked over time. However, this approach assumed that traffic signs

could be naively approximated using rectangular shapes, which might limit its generalization in complex urban environments.

2.4.2 Road Marking Primitive Mapping

Road marking primitives, such as dashed lines, solid lines, arrows, and curbs, are valuable features for autonomous driving, as they provide critical information about road structure and driving rules.

One of the earliest studies utilizing road markings for vehicle localization was presented by Pink et al. [Pin09]. A localization map consisting of road-marking pixels was generated from aerial imagery. Given an initial guess, the system tracked vehicle position using only image data. Similarly, LaneLoc, introduced by Schreiber et al. [Sch13], was a lane marking-based localization system. Its map was manually labeled using LiDAR point clouds projected into the Bird's Eye View (BEV) domain. During localization, a stereo camera system detected lane markings, which were matched with pre-labeled map features. A complementary approach by Schreiber et al. [Sch14] extended lane marking detection to include symbols such as arrows, stop signs, and speed limits. These symbols were identified and mapped using an Optical Character Recognition (OCR) technique applied to stereo camera point clouds projected in the BEV domain. The approaches discussed above detected road markings directly from fused point clouds without any tracking mechanisms.

Lanelet2, introduced by Poggenhans et al. [Pog18], is an open-source framework designed for HD-map representation. It provided a semantic map format that encodes road networks, including lanes, junctions, traffic signs, and other critical elements. By incorporating semantic information, the Lanelet2 framework facilitates localization, decision-making, and trajectory planning.

A more automated approach for road-marking detection and modeling from point clouds was proposed by Mi et al. [Mi21]. This approach employed YOLOv3 [Red18] to predict road-marking bounding boxes with semantic labels. Subsequently, a shape matching technique was applied to refine precise locations, orientations, and scales of road markings within these candidate

regions. Similarly, Yao et al. [Yao21] proposed a method for extracting, classifying, and vectorizing road markings. In this approach, semantic images were utilized to extract road-marking pixels, and corresponding 3D point clouds were retrieved. After classification into categories such as boundary lines and rectangular road markings, detected features were vectorized by aligning point cloud with matched model. However, these template-based approaches relied on predefined road-marking shapes, making them less generalizable to diverse road-marking conventions and variations.

Several works aim to develop more generalized road-marking modeling and mapping techniques. For instance, Prochazka et al. [Pro19] proposed a novel method for extracting and vectorizing lane markings. It detected road markings based on LiDAR reflectance values and represented them using a spanning tree structure. The vectorization process involved encapsulating each detected marking within a polygon envelope, offering a precise and detailed representation. However, while this method effectively captured the approximate spatial extent of road markings, it did not fully preserve exact contour shapes. Another method, introduced by Liu et al. [Liu20], extracted 3D vectors of road markings from point clouds using a hybrid approach that combined templates and topology. Landmarks were vectorized using optimized ICP and incremental convex hull construction. However, template-matching approaches, such as those used for complex shapes like arrows, were often limited in generalization due to variability in road-marking conventions.

In summary, existing semantic primitive mapping approaches share several common limitations across both elevated and road-marking primitives. Many methods rely on purely geometric characteristics without incorporating semantic information, while others depend on predefined templates or shape models that restrict generalizability across diverse real-world conditions. Parameter estimation is frequently performed on a per-frame basis without robust cross-frame tracking mechanisms, making the resulting maps susceptible to noise and temporal inconsistencies. Furthermore, few approaches tightly couple multi-modal sensor fusion with probabilistic parametric modeling to produce compact yet accurate map representations. In contrast, the semantic parametric

mapping system proposed in this thesis addresses these limitations in a unified framework for both elevated primitives, including poles, traffic lights, and traffic signs, and road-marking primitives, such as dashed lines, solid lines, arrows, and curbs. It combines instance segmentation masks from camera images with LiDAR point clouds to enable category-aware detection without reliance on rigid templates. A dedicated data association module tracks detected instances across frames, and their parametric representations are jointly optimized in a probabilistic framework, yielding improved robustness, generalization, and map compactness for all primitive types.

3 Fundamentals

3.1 Camera Models

A camera model establishes a mapping between 3D world space and the 2D image plane. A camera model mathematically defines this transformation, with the forward model, $\mathcal{P}_{\mathbf{F}} : \mathbb{R}^3 \to \mathbb{R}^2$, projecting 3D world points onto the 2D image plane. Conversely, the backward model, $\mathcal{P}_{\mathbf{B}} : \mathbb{R}^2 \to \mathbb{R}^3$, maps 2D image points back to 3D space. Unlike the forward model, the backward model does not provide a unique solution, as depth information is lost during projection. Instead, it maps 2D image points to viewing rays in 3D space along which the original 3D points lie. The choice of a camera model depends on application-specific requirements, such as field of view, distortion handling, mapping precision, and computational efficiency. Single View Point (SVP) camera models are a class of camera models that assume a single viewpoint for the entire image, meaning all light rays entering the camera converge at a single point. In this thesis, two commonly used SVP camera models are considered: the pinhole camera model and the spherical camera model.

3.1.1 Pinhole Camera Model

The pinhole camera model is the simplest and most widely adopted camera model. It is based on the principle that light travels in straight lines, projecting 3D world points onto a flat image plane. The camera coordinate system is defined by an orthogonal X-Y-Z frame, where the camera center is positioned at the origin. The optical axis aligns with the Z-axis, also referred to as the principal axis. The image plane is parallel to the X-Y plane, and the principal point $\mathbf{u}_\mathrm{c} = (u_\mathrm{c}, v_\mathrm{c})^\mathrm{T}$ is the intersection of the optical axis with the image

plane. The focal length is denoted as f. A 3D point $\mathbf{p} = (x, y, z)^{\mathrm{T}}$ in 3D world is projected onto a 2D image point with coordinates $\mathbf{u} = (u, v)^{\mathrm{T}}$, represented in homogeneous coordinates as $\breve{\mathbf{u}} = (u, v, 1)^{\mathrm{T}}$. By applying basic geometric principles, the following relationship is obtained:

$$z = f\frac{x}{u - u_{\mathrm{c}}} = f\frac{y}{v - v_{\mathrm{c}}}, \tag{3.1}$$

which can be rewritten in matrix form as the forward camera model:

$$\mathscr{P}_{\mathbf{F}}(x, y, z) = \breve{\mathbf{u}} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z}\begin{pmatrix} f & 0 & u_{\mathrm{c}} \\ 0 & f & v_{\mathrm{c}} \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{z}\mathbf{K}\mathbf{p}. \tag{3.2}$$

Here, $\mathbf{p}$ represents the 3D point in the world, and $\mathbf{K}$ is the camera intrinsic matrix. To obtain the 3D viewing ray from a given 2D image point in homogeneous coordinates $\breve{\mathbf{u}}$, the backward camera model is formulated as:

$$\mathscr{P}_{\mathbf{B}}(u, v) = \mathbf{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = z\mathbf{K}^{-1}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = z\mathbf{K}^{-1}\breve{\mathbf{u}}. \tag{3.3}$$

Since the depth z remains unknown, the backward model does not provide a unique 3D point. Instead, it defines a 3D viewing ray ρ characterized by a support point ρ_{s} and a direction vector $\rho_{\mathrm{d}} = \mathbf{K}^{-1}\breve{\mathbf{u}}$. The most common choice for the support point ρ_{s} is the origin of the camera coordinate system.

3.1.2 Spherical Camera Model

The spherical camera model is employed for cameras with wide fields of view, such as fisheye cameras. It overcomes the limitations of the pinhole camera model, which struggles with accurate image projections as the field of view approaches 180 degrees. The model maps 3D Cartesian coordinates to a 2D spherical coordinate system, effectively handling large distortions while providing a more accurate representation of image projections in wide-angle scenarios.

A 3D point $\mathbf{p}$ in the world is represented in spherical coordinates using the azimuth angle ϕ and the elevation angle θ, which define the corresponding 2D image point $\mathbf{s} = (\phi, \theta)^{\mathrm{T}}$ on the spherical surface. The transformation from Cartesian to spherical coordinates is given by:

$$\mathbf{s} = \begin{pmatrix} \phi \\ \theta \end{pmatrix} = \begin{pmatrix} \arctan 2(y, x) \\ \arctan 2\left(\sqrt{x^2 + y^2}, z\right) \end{pmatrix}. \tag{3.4}$$

To obtain final image coordinates, the projection follows the equidistance projection model. The forward camera model is thus defined as:

$$\mathscr{P}_{\mathbf{F}}(x, y, z) = \breve{\mathbf{u}} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f(-\phi) + u_{\mathrm{c}} \\ f\left(\theta - \frac{\pi}{2}\right) + v_{\mathrm{c}} \\ 1 \end{pmatrix} \tag{3.5}$$

By rearranging the forward camera model (3.5), the backward camera model can be formulated as:

$$\mathscr{P}_{\mathbf{B}}(u, v) = \mathbf{p}(\lambda) = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix} \lambda, \tag{3.6}$$

which maps the 2D image point $\mathbf{u}$ to a viewing ray in 3D world space. $\lambda \in \mathbb{R}^+$ denotes the Euclidean distance from the camera coordinate system origin to the 3D point $\mathbf{p}$ along the viewing ray.

3.2 3D Isometric Transformation

The 3D isometric transformation characterizes the rigid-body transformation between two 3D coordinate frames, a concept extensively applied in robotics and autonomous driving. A 3D isometric transformation:

$$\mathbf{T}_{\mathrm{A}}^{\mathrm{B}} \in \mathrm{SE}(3), \tag{3.7}$$

describes transformation from coordinate system $\mathcal{F}_A$ to coordinate system $\mathcal{F}_B$. Here, SE(3) represents the Special Euclidean group in 3D space, which models the motion of a rigid body within 3D space. A 3D isometric transformation can be decomposed into two components: the translation vector $\mathbf{t}_A^B \in \mathbb{R}^3$ and the rotation matrix $\mathbf{R}_A^B \in \mathrm{SO}(3)$. Here, SO(3) denotes the Special Orthogonal group in 3D space, representing all possible rotations in 3D. Composition of two 3D isometric transformations is expressed as:

$$\mathbf{T}_A^C = \mathbf{T}_B^C \mathbf{T}_A^B, \tag{3.8}$$

To transform a 3D point $\mathbf{p}_A \in \mathbb{R}^3$ from coordinate frame $\mathcal{F}_A$ to corresponding point $\mathbf{p}_B \in \mathbb{R}^3$ in coordinate frame $\mathcal{F}_B$, the following equation is used:

$$\mathbf{p}_B = \mathbf{T}_A^B \mathbf{p}_A. \tag{3.9}$$

3.3 Uniform B-spline

A B-spline is an extension of the Bézier curve. In order to understand B-splines, it is essential to first comprehend Bézier curves. A Bézier curve is defined as a weighted linear combination of n + 1 control points, $\mathbf{p}_0^c, \mathbf{p}_1^c, \ldots, \mathbf{p}_n^c$, with Bernstein polynomials as respective weights:

$$\mathbf{X}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{p}_i^c, \; 0 \leq t \leq 1, \tag{3.10}$$

where $(n, i)^T$ denotes the binomial coefficient and t is the independent parameter, which uniquely defines the function once control points are fixed. A Bézier curve of order $n + 1$ is defined by $n + 1$ control points and is a polynomial of degree n.

In a B-spline, each control point is associated with a corresponding basis function. In contrast to the Bézier curve shown in Equation (3.10), a B-spline can

be represented by the following expression:

$$\mathbf{S}(t) = \sum_{i=0}^{n} B_{\mathrm{i,k}}(t)\mathbf{p}_{\mathrm{i}}^{\mathrm{c}}, t_{\min} \leq t \leq t_{\max}, \tag{3.11}$$

where $B_{\mathrm{i,k}}(t)$ denotes the basis function/Bernstein polynomial of order $k+1$, and t is the range parameter. The non-decreasing sequence of real numbers $(t_0, t_1, \ldots, t_{\mathrm{m}})$, $t_{\mathrm{i}} \leq t_{\mathrm{i+1}}$, is referred to as the knot vector. For $\mathrm{k} = 0, 1, \ldots, n$, $k+1$-th order B-spline basis functions $B_{\mathrm{i,k}}(t)$ are recursively defined as follows:

$$B_{\mathrm{i},0}(t) = \begin{cases} 1, & t_{\mathrm{i}} \leq t < t_{\mathrm{i+1}} \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

$$B_{\mathrm{i,k}}(t) = \frac{(t - t_{\mathrm{i}})B_{\mathrm{i,k-1}}(t)}{t_{\mathrm{i+k-1}} - t_{\mathrm{i}}} + \frac{(t_{\mathrm{i+k}} - t)B_{\mathrm{i+1,k-1}}(t)}{t_{\mathrm{i+k}} - t_{\mathrm{i+1}}}. \tag{3.13}$$

Bézier curves provide global control, meaning that all control points directly influence the shape of the curve. In contrast, B-splines offer more localized control, as each control point affects only a specific subset of the curve, facilitating the modification of a small section without altering the entire curve. Additionally, a Bézier curve must pass through its first and last control points, while a B-spline does not have this restriction.

A B-spline is called a uniform B-spline if knots are equally spaced, which significantly improves computational efficiency, as basis functions can be precomputed. Uniform B-spline basis functions retain most properties of general B-spline basis functions, such as non-negativity, local support, partition of unity, continuity order, linear independence, and symmetry.

3.4 Nonlinear Optimization

Many problems in robotics and autonomous driving require estimating unknown parameters, such as sensor poses or map features, from noisy measurements. In general, the number of measurements exceeds the number of unknowns, and the relationship between them is nonlinear. The goal is to find the parameter values that best explain the observed measurements, which is formulated as a nonlinear least-squares problem.

Let $\mathbf{x} \in \mathbb{R}^n$ denote the n-dimensional parameter vector to be estimated, and let $\mathbf{r}(\mathbf{x}) = \{r_1(\mathbf{x}), ..., r_\mathrm{m}(\mathbf{x})\}^\mathrm{T}$ represent the m-dimensional residual vector, where each residual $r_\mathrm{i}(\mathbf{x})$ measures the discrepancy between a predicted and an observed measurement. The optimization problem seeks the parameter vector $\mathbf{x}$ that minimizes the sum of squared residuals:

$$\arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|^2, \quad \mathbf{L} \leq \mathbf{x} \leq \mathbf{U}, \tag{3.14}$$

where $\mathbf{L}$ and $\mathbf{U}$ represent optional lower and upper bounds on $\mathbf{x}$, respectively. Since $\mathbf{r}(\mathbf{x})$ is generally nonlinear in $\mathbf{x}$, this problem cannot be solved in closed form and is instead solved iteratively.

Iterative linearization. At each iteration, the residual function is linearized around the current parameter estimate $\mathbf{x}$ using a first-order Taylor expansion:

$$\mathbf{r}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{r}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\,\Delta\mathbf{x}, \tag{3.15}$$

where $\mathbf{J}(\mathbf{x}) \in \mathbb{R}^{m \times n}$ is the Jacobian matrix containing the partial derivatives of each residual with respect to each parameter. Substituting this approximation into Equation (3.14) yields the linearized subproblem:

$$\arg\min_{\Delta\mathbf{x}} \frac{1}{2} \|\mathbf{r}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\Delta\mathbf{x}\|^2, \quad \mathbf{L} \leq \mathbf{x} + \Delta\mathbf{x} \leq \mathbf{U}, \tag{3.16}$$

which is a linear least-squares problem in $\Delta\mathbf{x}$ and can be solved efficiently. Setting the gradient of Equation (3.16) with respect to $\Delta\mathbf{x}$ to zero yields the

normal equations:

$$\underbrace{\mathbf{J}^{\mathrm{T}}(\mathbf{x})\,\mathbf{J}(\mathbf{x})}_{\approx \text{Hessian } \mathbf{H}}\,\Delta\mathbf{x} = -\mathbf{J}^{\mathrm{T}}(\mathbf{x})\,\mathbf{r}(\mathbf{x}). \tag{3.17}$$

The product $\mathbf{H} = \mathbf{J}^{\mathrm{T}}\mathbf{J}$ approximates the Hessian of the cost function and captures the curvature of the objective in parameter space. The right-hand side $-\mathbf{J}^{\mathrm{T}}\mathbf{r}(\mathbf{x})$ is the negative gradient of the cost, pointing in the direction of steepest descent. Update of the parameter vector $\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$ leads to various optimization methods. The most commonly used methods are the Gauss-Newton method and the Levenberg-Marquardt (LM) method. For a comprehensive treatment of nonlinear optimization theory, refer to Nocedal et al. [Noc06]. In this thesis, the Ceres Solver Agarwal et al. [Aga23] is employed to solve nonlinear optimization problems, as it provides efficient implementations of both LM and robust loss functions.

3.5 Pose Graph Optimization

Pose Graph Optimization (PGO) is a widely used backend framework in SLAM for globally consistent trajectory and map estimation. The core idea is to represent the estimation problem as a graph, where the unknown poses are nodes and sensor-derived constraints between them are edges. By optimizing over all nodes simultaneously, accumulated drift from sequential odometry can be corrected, yielding a globally consistent estimate.

Graph structure. In a PGO problem, a pose graph is constructed that comprises vertices and edges. Vertices represent poses $\hat{\mathbf{T}}_i \in \mathrm{SE}(3)$ to be estimated, typically associated with keyframes at discrete time steps. Edges encode constraints between poses, derived from various sensor sources:

- GNSS: provides absolute position measurements in a global coordinate system, anchoring the trajectory to a known reference frame.
- Odometry: provides relative motion estimates between consecutive poses, derived from sources such as IMU preintegration, wheel encoders,

or visual or LiDAR scan matching. These constitute the dominant source of edges in most SLAM systems.

- Loop closure: provides relative constraints between non-adjacent poses that correspond to the same physical location, visited at different times. Loop closures are critical for correcting long-term drift.

Constraint residuals. Each edge encodes a constraint as a residual between the measured and the estimated relative or absolute transformation. Two primary types of constraints are distinguished: absolute and relative. For an absolute constraint, such as a GNSS measurement $\tilde{\mathbf{T}}_i$ for pose $\hat{\mathbf{T}}_i$, the residual is:

$$\mathbf{r}_a = \tilde{\mathbf{T}}_i \ominus \hat{\mathbf{T}}_i, \tag{3.18}$$

where $\ominus$ denotes the pose difference operator on SE(3), returning the Lie algebra element (i.e., a 6-dimensional error vector) corresponding to the difference between the two poses. For a relative constraint between poses $\hat{\mathbf{T}}_i$ and $\hat{\mathbf{T}}_j$, the residual compares the measured relative transformation $\tilde{\mathbf{T}}_i^j$ against the relative transformation inferred from the estimated poses:

$$\mathbf{r}_r = \tilde{\mathbf{T}}_i^j \ominus \left(\hat{\mathbf{T}}_j^{-1} \hat{\mathbf{T}}_i \right), \tag{3.19}$$

where $\hat{\mathbf{T}}_j^{-1} \hat{\mathbf{T}}_i$ is the relative transformation from pose j to pose i as implied by the current estimates. A non-zero residual indicates a discrepancy between the measurement and the current pose estimates that the optimizer will seek to reduce.

Role of loop closure. Without loop closure, small per-step odometric errors accumulate over time, causing the trajectory to drift and resulting in globally inconsistent maps. Loop closure detection identifies previously visited locations and introduces constraints between non-adjacent poses. These constraints allow the optimizer to distribute accumulated drift across the entire trajectory, yielding a globally consistent estimate.

4 Continuous Time SLAM

As discussed in Section 2.1, Section 2.2, and Section 2.3, the most commonly used Simultaneous Localization and Mapping (SLAM) optimization backend method is the pose graph optimization (PGO) algorithm. However, a pose graph is constructed using time-discrete poses, and poses are estimated by optimizing the pose graph. This time discretization presents challenges in utilizing time-asynchronous measurements and measurements with varying acquisition rates. Moreover, it becomes difficult to estimate velocities and accelerations in a smooth and continuous manner from time-discrete poses. Another disadvantage of the PGO algorithm is that the number of parameters to be optimized is proportional to the number of poses, which increases both the complexity and computational cost of the optimization problem.

To address these shortcomings, this thesis employs a continuous-time trajectory representation as an optimization backend in the SLAM system. This representation possesses the following key properties:

- **Time Continuity and Differentiability:** It ensures time continuity to model motion, along with first- and second-order differentiability to implicitly model velocities and accelerations.
- **Reasonable Parameterization:** A reasonable parameterization is employed to ensure usability in the optimization process while minimizing the complexity of parameterization.
- **Local Support:** Measurements should only influence motion estimation within a local range where they have physical relevance.

In this chapter, a continuous-time SLAM approach is presented, which utilizes a continuous-time trajectory representation as an optimization backend

method. Based on existing literature, B-spline curves are identified as an appropriate method for this purpose. To manage complexity, uniform B-splines are used to model continuous-time trajectory. Given that translation and rotation components of a 3D isometric transformation are physically independent and can be parameterized separately, two uniform B-splines are employed to model continuous-time trajectory: one for translation $\mathbf{t}(t) = \mathbf{S}_{\mathbf{t}}(t)$ and one for rotation $\mathbf{R}(t) = \mathbf{S}_{\mathbf{R}}(t)$.

This uniform B-spline-parameterized trajectory allows for smooth and continuous estimation of motion poses, as well as linear and angular velocities and accelerations. Conversely, this representation also enables the construction of accurate and effective measurement models to estimate the trajectory from pose, velocity, and acceleration measurements.

The structure of this chapter is as follows: Spatial rotation parameterization methods are discussed in Section 4.1.1. A uniform B-spline-based continuous-time trajectory, as an optimization backend method for SLAM, is introduced in Section 4.1.2. Time derivatives of uniform B-splines, which are introduced in Section 4.1.3, are used to estimate linear and angular velocities and accelerations from the continuous-time trajectory. The axis-angle representation, selected as the rotation parameterization method, and the conversion between axis-angle representation and rotation matrix are discussed in Section 4.1.4. The time derivative of the axis-angle representation is discussed in Section 4.1.4 as one of the main contributions of this thesis. Different sensor measurement types, including GNSS, IMU, camera, and LiDAR, are introduced in Section 4.2, and measurement models for these sensors are discussed. A LiDAR point cloud-based loop-closure method, presented in Section 4.3, is used to close loops and correct for drift in the continuous-time trajectory from bundle adjustment motion estimation. Based on continuous-time trajectory theory and measurement models for various sensors, the continuous-time bundle adjustment algorithm is presented in Section 4.4.

4.1 Continuous-Time Trajectory

4.1.1 3D Isometric Transformation Parameterization

As outlined in Section 3.2, a 3D isometric transformation $\mathbf{T}_{\mathrm{A}}^{\mathrm{B}}$ consists of two components: the translation $\mathbf{t}_{\mathrm{A}}^{\mathrm{B}}$ and the rotation $\mathbf{R}_{\mathrm{A}}^{\mathrm{B}}$. The translation $\mathbf{t}_{\mathrm{A}}^{\mathrm{B}}$ represents the displacement of the origin of the coordinate system and can be easily parameterized as a 3D vector in Euclidean space:

$$\mathbf{t}_{\mathrm{A}}^{\mathrm{B}} = (x, y, z)^{\mathrm{T}}. \tag{4.1}$$

A 3D point $\mathbf{p}_{\mathrm{A}}$ in coordinate system $\mathcal{F}_{\mathrm{A}}$ can be translated to point $\mathbf{p}_{\mathrm{B}}$ in coordinate system $\mathcal{F}_{\mathrm{B}}$ using the equation:

$$\mathbf{p}_{\mathrm{B}} = \mathbf{p}_{\mathrm{A}} + \mathbf{t}_{\mathrm{A}}^{\mathrm{B}}. \tag{4.2}$$

In contrast to translation, rotation is more complex and can be parameterized in various forms, such as rotation matrices, Euler angles, quaternions, and axis-angle representations. Each of these methods has its advantages and disadvantages, which are discussed in the following subsections.

Rotation Matrix

A rotation matrix $\mathbf{R}_{\mathrm{A}}^{\mathrm{B}}$ is a 3×3 matrix that transforms a vector from coordinate system $\mathcal{F}_{\mathrm{A}}$ to coordinate system $\mathcal{F}_{\mathrm{B}}$. In a geometric interpretation, it describes the orientation of coordinate system $\mathcal{F}_{\mathrm{A}}$ relative to $\mathcal{F}_{\mathrm{B}}$. Using the rotation matrix, a 3D point $\mathbf{p}_{\mathrm{A}}$ in coordinate system $\mathcal{F}_{\mathrm{A}}$ can be rotated to point $\mathbf{p}_{\mathrm{B}}$ in coordinate system $\mathcal{F}_{\mathrm{B}}$ according to the equation:

$$\mathbf{p}_{\mathrm{B}} = \mathbf{R}_{\mathrm{A}}^{\mathrm{B}} \mathbf{p}_{\mathrm{A}}. \tag{4.3}$$

The rotation matrix is a special orthogonal matrix, and the set of all 3×3 rotation matrices forms the Special Orthogonal group SO(3):

$$\mathrm{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} \mid \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}_{\mathrm{d}}, \det(\mathbf{R}) = 1\}. \tag{4.4}$$

A rotation matrix is straightforward to interpret, and point transformation is simple to compute. However, since a 3×3 matrix with 9 parameters is used to describe a rotation with three Degrees of Freedom (DoF), it is redundant. Moreover, the matrix must satisfy the constraints of the Special Orthogonal group SO(3), which increases the complexity of optimization problems.

Euler Angles

Euler angles define rotations about three axes X, Y, and Z of a fixed coordinate system. However, the order in which rotations are applied is not unique, leading to different conventions. For instance, in aerospace applications, the roll-pitch-yaw convention is commonly used, where roll corresponds to rotation about the X-axis, pitch about the Y-axis, and yaw about the Z-axis. The corresponding rotation matrix is then calculated as:

$$\mathbf{R} = \mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha), \tag{4.5}$$

where $\mathbf{R}_z(\gamma)$, $\mathbf{R}_y(\beta)$, and $\mathbf{R}_x(\alpha)$ represent rotation matrices around the Z-, Y-, and X-axes, respectively. Euler angles are relatively easy to understand and interpret, requiring three parameters to describe a rotation with three DoF without redundancy. However, they suffer from issues such as non-uniqueness in axis rotation order and the gimbal-lock problem, which occurs when two axes become aligned, leading to loss of one DoF. As a result, Euler angles are primarily used in specific applications where the rotation convention is fixed and the gimbal-lock issue can be avoided. For general 3D rotations, Euler angles are less suited for optimization tasks.

Quaternion

Quaternions provide a more compact representation of 3D rotation than rotation matrices and are free from singularities, unlike Euler angles. The main drawback is the lack of intuitive interpretation. A quaternion $\mathbf{q}$ consists of a real part q_w and an imaginary part $\mathbf{q}_v = (q_x, q_y, q_z)^{\mathrm{T}}$:

$$\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}, \tag{4.6}$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the imaginary units. For a 3D point $\mathbf{p}$ and quaternion $\mathbf{q}$, the rotated point $\mathbf{p}'$ is computed as:

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}. \tag{4.7}$$

A quaternion uses four parameters to describe a rotation with three DoF, resulting in redundancy. The four parameters must satisfy the unit-norm constraint: $\mathbf{q}^{\mathrm{T}}\mathbf{q} = 1$, which adds complexity to the optimization process. To address this, an additional local parameterization of the quaternion is introduced during optimization to ensure that the unit-norm constraint is maintained.

Axis-Angle

Axis-angle representation is a compact and singularity-free method for describing rotations. The axis-angle representation $\varphi = \psi\mathbf{e}$ consists of two parts: the unit vector $\mathbf{e}$ representing the rotation axis and the angle ψ representing the rotation about $\mathbf{e}$. To convert from axis-angle to rotation matrix, Rodrigues' rotation formula is used:

$$\mathbf{R} = \mathbf{I}_{\mathrm{d}} + \sin(\psi)\mathbf{A} + (1 - \cos(\psi))\mathbf{A}^2, \tag{4.8}$$

where $\mathbf{A}$ is the skew-symmetric matrix of the unit vector $\mathbf{e} = (e_x, e_y, e_z)^{\mathrm{T}}$:

$$\mathbf{A} = \begin{pmatrix} 0 & -e_z & e_y \\ e_z & 0 & -e_x \\ -e_y & e_x & 0 \end{pmatrix}. \tag{4.9}$$

Axis-angle representation requires three parameters to describe a rotation with three DoF, with no redundancy. Unlike rotation matrix and quaternion representations, it does not impose any implicit constraints. These attributes make the axis-angle representation particularly well-suited for optimization tasks, such as bundle adjustment and PGO problems.

4.1.2 Continuous-Time Trajectory Representation

Based on the advantages and disadvantages of the various 3D rotation parameterization methods discussed in Section 4.1.1, axis-angle is chosen as the rotation parameterization for modeling the continuous-time trajectory.

Each axis-angle coordinate φ represents a unique rotational movement, with every rotation having a corresponding distinct axis-angle representation. This property makes the axis-angle method highly suitable for continuous-time trajectory modeling. Moreover, continuous rotational movement is also continuous in axis-angle space, enabling an excellent approximation of real-world motion when combined with continuous uniform B-splines.

Additionally, using Rodrigues' formula Equation (4.10), rotational movements can be easily determined from axis-angle vectors. Through the time derivatives of uniform B-splines in the axis-angle representation, both angular velocity and acceleration can be estimated. Rodrigues' formula provides an efficient algorithm to compute the exponential map from the Lie algebra so(3) to the rotation group SO(3):

$$\mathbf{R} = \mathbf{I}_\mathrm{d} + \sin\psi[\mathbf{e}]_\times + (1-\cos\psi)[\mathbf{e}]_\times^2, \tag{4.10}$$

where

$$[\mathbf{e}]_\times := \begin{pmatrix} 0 & -\varphi_3 & \varphi_2 \\ \varphi_3 & 0 & -\varphi_1 \\ -\varphi_2 & \varphi_1 & 0 \end{pmatrix} \in \mathrm{Skew}_3 \tag{4.11}$$

is the skew-symmetric matrix of vector $\mathbf{e}$, such that $[\mathbf{e}]_\times\mathbf{b} = \mathbf{e}\times\mathbf{b}$ for all vectors $\mathbf{e} = (\varphi_1, \varphi_2, \varphi_3)^\mathrm{T}$ and $\mathbf{b} \in \mathbb{R}^3$. The matrix $\mathbf{I}_\mathrm{d}$ represents the identity matrix.

The axis-angle representation $\boldsymbol{\varphi} := \psi \mathbf{e}$ is a compact and natural representation of rotation described in terms of its geometric components. Rodrigues' formula, Equation (4.10), is a closed-form expression of the exponential map:

$$\mathbf{R} = \exp([\boldsymbol{\varphi}]_\times) := \sum_{k=0}^{\infty} \frac{1}{k!} [\boldsymbol{\varphi}]_\times^k = \sum_{k=0}^{\infty} \frac{\psi^k}{k!} [\mathbf{e}]_\times^k. \tag{4.12}$$

4.1.3 Derivatives of Uniform B-Splines

Uniform B-splines are utilized as an optimization backend in SLAM to incorporate pose, velocity, and acceleration measurements, requiring first and second derivatives of uniform B-splines. To derive uniform B-splines $\mathbf{S}(t)$, it suffices to compute the derivatives of the basis functions $B_{\mathrm{i,k}}(t)$, as the control points are constants. Based on the definition of the basis functions in Equation (3.13), the computation of first- and second-order time derivatives of the basis functions can be summarized by the following two equations:

$$\frac{\mathrm{d}B_{\mathrm{i,k}}(t)}{\mathrm{d}t} = \frac{B_{\mathrm{i,k-1}}(t)}{(t_{\mathrm{i+k-1}} - t_{\mathrm{i}})} - \frac{B_{\mathrm{i+1,k-1}}(t)}{(t_{\mathrm{i+k}} - t_{\mathrm{i+1}})} \tag{4.13}$$

$$\frac{\mathrm{d}^2 B_{\mathrm{i,k}}(t)}{\mathrm{d}^2 t} = \frac{\mathrm{d}B_{\mathrm{i,k-1}}(t)}{(t_{\mathrm{i+k-1}} - t_{\mathrm{i}})\mathrm{d}t} - \frac{\mathrm{d}B_{\mathrm{i+1,k-1}}(t)}{(t_{\mathrm{i+k}} - t_{\mathrm{i+1}})\mathrm{d}t}, \tag{4.14}$$

where i is the index of control points, k is the order of the uniform B-spline, t is the time variable, and t_{i} is the i-th time knot.

The order of the uniform B-splines must be selected appropriately. To make use of all measurements, including accelerations, in the SLAM system, uniform B-splines should be second-order time-differentiable. To satisfy this condition, uniform cubic B-splines are chosen, implying that the parameter k in Equation (4.13) and Equation (4.14) should be set to 3.

4.1.4 Time Derivatives of a 3D Rotation

As described in Section 4.1.2, to estimate velocities and accelerations from a uniform B-spline-modeled continuous-time trajectory, not only the time derivatives of the uniform B-spline but also the time derivatives of the rotation representation are required. For the translation component, linear velocities can be estimated as: $\mathbf{v} = \frac{\mathrm{d}\mathbf{t}}{\mathrm{d}t}$ and linear accelerations as: $\mathbf{a} = \frac{\mathrm{d}^2\mathbf{t}}{\mathrm{d}t^2}$ from the translation coordinate vector $\mathbf{t} = (x, y, z)^{\mathrm{T}}$.

In contrast to the straightforward estimation of linear velocity $\mathbf{v}$ and acceleration $\mathbf{a}$, the estimation of angular velocity ω and angular acceleration $\dot{\omega}$ with respect to the axis-angle φ is more complex and computationally intensive. Although formulas exist to express the derivative of the exponential map in general Lie groups [Hel62, Hal10], this thesis uses the analytical formula from Gallego et al. [Gal15], which computes the time derivatives of the rotation matrix $\mathbf{R}(\varphi)$ with respect to the axis-angle φ in a relatively simple manner. The derivative of $\mathbf{R}(\varphi) = \exp([\varphi]_\times)$, as described in Equation (4.12), with respect to its axis-angle coordinates $\varphi = (\varphi_1, \varphi_2, \varphi_3)^{\mathrm{T}}$ can be formulated as:

$$\frac{\partial \mathbf{R}}{\partial \varphi_\mathrm{i}} = \frac{\varphi_\mathrm{i}[\varphi]_\times + [\varphi \times (\mathbf{I}_\mathrm{d} - \mathbf{R})\mathbf{b}_\mathrm{i}]_\times}{\|\varphi\|^2}\mathbf{R}, \tag{4.15}$$

where $\mathbf{b}_\mathrm{i}$ denotes the i-th vector of the standard basis in $\mathbb{R}^3$, and $\mathbf{I}_\mathrm{d}$ is the 3×3 identity matrix. As $\varphi \to \mathbf{0}$, Equation (4.15) cannot be used directly, requiring approximations. Following Gallego et al. [Gal15], by making use of the facts that $\lim_{\varphi\to\mathbf{0}} \mathbf{R} = \mathbf{I}_\mathrm{d}$ and $\lim_{\varphi\to\mathbf{0}} \frac{\mathbf{I}_\mathrm{d}-\mathbf{R}}{\|\varphi\|} = -[\mathbf{e}]_\times$, the derivative at the identity can be derived by computing the limit as $\varphi \to \mathbf{0}$:

$$\begin{aligned}
\lim_{\varphi\to\mathbf{0}} \frac{\partial \mathbf{R}}{\partial \varphi_\mathrm{i}} &= \lim_{\varphi\to\mathbf{0}} \left(\frac{\varphi_\mathrm{i}[\mathbf{e}]_\times + [\mathbf{e} \times (\mathbf{I}_\mathrm{d} - \mathbf{R})\mathbf{b}_\mathrm{i}]_\times}{\|\varphi\|^2}\mathbf{R} \right) \\
&= \varphi_\mathrm{i}[\mathbf{e}]_\times - [\mathbf{e} \times ([\mathbf{e}]_\times \mathbf{b}_\mathrm{i})]_\times \\
&= [\varphi_\mathrm{i}\mathbf{e} - [\mathbf{e}]_\times^2 \mathbf{b}_\mathrm{i}]_\times \\
&= [\mathbf{b}_\mathrm{i}]_\times.
\end{aligned} \tag{4.16}$$

Using the axis-angle φ and uniform B-splines in Equation (3.11), the first- and second-order time derivatives $\frac{\mathrm{d}\varphi}{\mathrm{d}t}$ and $\frac{\mathrm{d}^2\varphi}{\mathrm{d}t^2}$ can be calculated from the rotation uniform B-spline $\mathbf{S}_{\mathbf{R}}$. These derivatives, combined with Equation (4.13) and Equation (4.14), enable the estimation of angular velocity and acceleration. The angular velocity ω can be computed as:

$$\omega = \frac{\mathrm{d}\mathbf{S}_{\mathbf{R}}}{\mathrm{d}t} = \sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\mathrm{d}B_{\mathrm{i,k}}(t)}{\mathrm{d}t}\mathbf{p}_{\mathrm{i}}^{\mathrm{c}} \tag{4.17}$$

The angular acceleration $\dot{\omega}$ is given by:

$$\dot{\omega} = \frac{\mathrm{d}^2\mathbf{S}_{\mathbf{R}}}{\mathrm{d}t^2} = \sum_{\mathrm{i}=0}^{\mathrm{n}} \frac{\mathrm{d}^2B_{\mathrm{i,k}}(t)}{\mathrm{d}t^2}\mathbf{p}_{\mathrm{i}}^{\mathrm{c}}, \tag{4.18}$$

where $\mathbf{p}_{\mathrm{i}}^{\mathrm{c}}$ are control points of the uniform B-spline.

Additionally, the relationship between the first-order derivative of the rotation matrix $\mathbf{R}$ and the skew-symmetric matrix of the angular velocity ω can be expressed as:

$$[\omega]_{\times} = \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\mathbf{R}^{-1} = \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\mathbf{R}^{\mathrm{T}}, \tag{4.19}$$

which allows the angular velocity ω to be easily computed from the rotation matrix $\mathbf{R}$ and its first-order time derivative $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}$. The angular acceleration $\dot{\omega}$ can be calculated from the rotation matrix $\mathbf{R}$, its first-order time derivative $\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}$, and its second-order time derivative $\frac{\mathrm{d}^2\mathbf{R}}{\mathrm{d}t^2}$ using the following equation:

$$[\dot{\omega}]_{\times} = \frac{\mathrm{d}^2\mathbf{R}}{\mathrm{d}t^2}\mathbf{R}^{\mathrm{T}} + \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\left(\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\right)^{\mathrm{T}}. \tag{4.20}$$

By combining Equation (4.15) with Equation (4.19), the angular velocity ω can be calculated from the following equation:

$$[\omega]_\times = \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\mathbf{R}^\mathrm{T} = \left[\frac{\partial \mathbf{R}}{\partial \varphi_1}, \frac{\partial \mathbf{R}}{\partial \varphi_2}, \frac{\partial \mathbf{R}}{\partial \varphi_3}\right] \frac{\mathrm{d}\varphi}{\mathrm{d}t}\mathbf{R}^\mathrm{T}. \tag{4.21}$$

Now, the angular velocity ω is derived as a function of the rotation matrix $\mathbf{R}$ and axis-angle coordinates φ, which can be easily estimated from the uniform B-spline $\mathbf{S}_\mathbf{R}$. As the main contribution, Equation (4.21) serves as a measurement model for angular velocity measurements from IMU sensors. Since no sensors are used in this thesis to measure angular acceleration, Equation (4.20) is not utilized to create a corresponding measurement model.

4.2 Measurements

After introducing the uniform B-spline-based continuous-time trajectory SLAM optimization backend, the following sensor measurements, which are used in the SLAM algorithm for this thesis, are presented:

- **GNSS**: Global positioning information.
- **IMU**: Linear acceleration and angular velocity.
- **Camera**: 2D image feature-point matches.
- **LiDAR**: 3D point cloud voxel feature matches.

4.2.1 Global Navigation Satellite System

GNSS is a satellite-based system that provides global positioning data. The basic principle behind GNSS systems is to calculate the distance between satellites and receivers by measuring the time difference between when signals are sent by satellites and when they are received by receivers. Using distance information from at least four satellites, receivers can determine their position.

GNSS sensors are typically affordable and robust, making them suitable for use in various weather and lighting conditions. Due to their ability to provide global position, GNSS sensors are integrated into SLAM systems to geo-reference mapped local areas to the global coordinate system. However, their accuracy

is limited, particularly in urban or forested areas where buildings or trees can obstruct signals. In extreme cases, such as in tunnels, GNSS signals may be completely unavailable. Because of their limited accuracy, GNSS measurements are not used as primary inputs but rather as reference measurements for globally adjusting mapped areas in SLAM systems.

4.2.2 Inertial Measurement Unit

IMU sensors typically consist of three accelerometers and three gyroscopes, enabling them to measure motion in 3D space. Unlike GNSS sensors, IMU sensors do not rely on external signals, which makes them valuable in environments where GNSS signals are unavailable, such as tunnels. IMU sensors work by measuring the linear acceleration and angular velocity of objects using their inertia. The measurements provided by IMU sensors are linear acceleration $\mathbf{a} = (a_x, a_y, a_z)^T$ and angular velocity $\omega = (\omega_x, \omega_y, \omega_z)^T$, both of which are expressed in the sensor-fixed local coordinate system.

IMU sensors are known for their robustness, making them suitable for use in demanding environments. Additionally, the high measurement frequency of IMU sensors allows for real-time applications, such as in drones, robots, or autonomous vehicles. However, to incorporate IMU measurements into SLAM systems, they need to be integrated into translation and rotation measurements. During this integration process, errors accumulate, which can lead to a gradual decrease in measurement accuracy over time.

In this thesis, IMU measurements are directly used, leveraging the uniform B-spline-based continuous-time trajectory optimization backend. The high measurement frequency contributes to creating smooth and optimal trajectories in continuous time, enhancing the overall performance of SLAM systems.

4.2.3 Camera Feature Points

Camera sensors are among the most essential sensors in computer vision and SLAM systems. Like GNSS and IMU sensors, cameras are passive sensors,

meaning they do not emit any signals to the external environment. The mapping process from 3D world to 2D images is discussed in Section 3.1. However, unlike GNSS and IMU sensors, cameras capture significantly more information about the environment through 2D image pixels. By applying specific detectors and descriptors, distinctive 2D image points can be identified and matched across different images, which can then be used to estimate both camera poses and 3D point coordinates simultaneously. These image points are referred to as feature points or keypoints, and methods for detecting and describing them are known as feature point detectors and feature point descriptors, respectively.

Numerous feature point detectors are available, such as the Harris corner detector, the FAST corner detector, the SIFT detector, the SURF detector, and the ORB detector. In this thesis, images are discretely convolved with blob and corner detection masks to detect feature points. Once feature points are detected, a descriptor is applied to characterize each point, and these descriptors are used to match feature points between different images. Many feature descriptors exist, including SIFT, SURF, ORB, and BRIEF descriptors. For this thesis, DIRD is employed as the feature descriptor. The DIRD descriptor is a learned combination of multiple descriptors, providing robust and reliable representation of feature points.

The process of matching feature points across images, also known as the association process, involves finding corresponding feature points in different frames. This matching is typically performed by comparing the feature descriptors of detected points. Several distance metrics can be used for comparison, such as Euclidean distance, Mahalanobis distance, Hamming distance, or cosine similarity. To correctly identify corresponding feature points, the distance between feature descriptors should be minimized. Based on the feature-point matching results, feature-point tracklets can be constructed. A feature-point tracklet is a set of associated feature points, each detected in different images but corresponding to the same 3D point in the real world. Using the estimated feature-point tracklets, a residual cost function can be constructed to jointly estimate the camera poses and 3D point coordinates.

Feature-point-based methods are chosen over direct methods in this thesis for several reasons. Feature points provide a compact scene representation, reducing optimization complexity compared to dense pixel-based methods. Feature descriptors are robust to photometric and viewpoint changes, and the reprojection error yields a geometrically well-defined cost that enables efficient sparse optimization. In contrast, direct methods rely on photometric consistency, an assumption frequently violated in outdoor driving due to changing illumination and fisheye camera vignetting.

Residual Construction and Pose Optimization

As described in Section 3.4, the estimation of camera poses and 3D point coordinates is formulated as a nonlinear optimization problem. The residual cost function used in this thesis is the reprojection error, which involves the camera models. To solve this nonlinear optimization problem, the LM method, outlined in Section 3.4, is applied:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|_{\Omega}^{2}, \tag{4.22}$$

where $\hat{\mathbf{x}}$ is the optimized parameter vector, including both the camera poses and 3D point coordinates, $\mathbf{r}(\mathbf{x})$ is the reprojection error residual, and Ω is the information matrix, which weights the residuals.

A commonly used approach to estimate camera poses and 3D point coordinates from associated feature-point tracklets is bundle adjustment. A landmark $\mathbf{l}_j$ in 3D space corresponds to a set of image feature points, $\mathcal{Z}_i = \{\mathbf{z}_1, \dots, \mathbf{z}_i, \dots, \mathbf{z}_n\}$. The image point $\mathbf{z}_i$ originates from the image $\mathbf{I}_i$, which was captured by the camera with intrinsic matrix $\mathbf{K}_i$. The camera pose at the time the image $\mathbf{I}_i$ was taken is denoted as $\mathbf{T}_i$. Thus, the reprojection error for each feature point in the image can be expressed as:

$$\mathbf{r}_{\mathbf{z}_i} = \mathbf{z}_i - \pi_i(\mathbf{K}_i^{-1}\mathbf{T}_i^{-1}\mathbf{l}_j), \tag{4.23}$$

where $\mathbf{l}_j$ represents the 3D feature-point coordinates, and π_i is the projection function derived from the camera model. The reprojection error $\mathbf{r}_{\mathbf{z}_i}$ is the difference between the detected feature point $\mathbf{z}_i$ and the projected feature point, $\pi_i(\mathbf{K}_i^{-1}\mathbf{T}_i^{-1}\mathbf{l}_j)$.

The objective of the bundle adjustment algorithm is to minimize the reprojection error $\mathbf{r}_{\mathbf{z}_i}$ for all feature points $\mathbf{z}_i$. Combining the reprojection error from Equation (4.23) with the nonlinear optimization cost function in Equation (4.22), the bundle adjustment problem can be expressed as:

$$\hat{\mathcal{T}}, \hat{\mathcal{L}} = \arg\min_{\mathcal{T},\mathcal{L}} \sum_{j=1}^{|\mathcal{L}|} \sum_{\mathbf{z}_i \in \mathcal{Z}_j} \left\| \mathbf{r}_{\mathbf{z}_i} \right\|^2_{\Omega_{\mathbf{z}_i}} . \tag{4.24}$$

Here, $\hat{\mathcal{T}}$ represents the optimized set of camera poses, and $\hat{\mathcal{L}}$ is the set of optimized feature point coordinates. The goal is to minimize the reprojection error $\mathbf{r}_{\mathbf{z}_i}$ for all corresponding feature point sets $\mathcal{Z}_j$. Each reprojection error $\mathbf{r}_{\mathbf{z}_i}$ is associated with a weighting matrix $\Omega_{\mathbf{z}_i} \in \mathbb{R}^{2\times 2}$.

4.2.4 LiDAR Point Cloud

The LiDAR sensor is an active sensor that uses laser beams to measure the distance between the sensor and objects, making it suitable for use in low-light conditions, including at night. Unlike camera-based systems, which require at least two cameras with a known baseline to estimate the 3D position of feature points, the LiDAR sensor can directly provide depth information from a single scan. LiDAR sensors generate 3D point clouds that are highly accurate in terms of depth. To achieve results similar to image feature point detection, description, and matching, we introduce the process of point cloud voxelization and feature extraction in the following sections.

Voxelization

Voxelization is a process that discretizes 3D point clouds into 3D voxels, reducing their dimensionality and noise, thus making feature extraction more efficient and robust. In addition, the 3D world often contains regular structures such as edges and planes, which are typically represented by multiple points rather than a single point. This characteristic makes voxelization particularly effective for detecting such structures. To optimize the voxelization process, a tree data structure, such as an octree, is commonly used. With an octree, voxel sizes can be multi-scale and adaptively adjusted based on the distribution of points within each voxel, improving both efficiency and accuracy.

Feature Extraction and Association

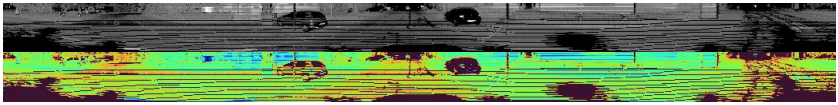
Before feature extraction can take place, ground points must be identified and separated from the rest of the point cloud, as they form a structurally distinct surface category that would otherwise interfere with edge and plane feature fitting. From the remaining non-ground points, two main types of features are then extracted: edge features and plane features. Following the LOAM framework [Zha14], points are classified solely by local surface curvature – high-curvature points form edge features, while low-curvature points form plane features – as these two categories are geometrically stable, widely present in structured environments, and together sufficient to constrain all six degrees of freedom of the sensor pose.

To extract ground points, we use the ground surface estimation method proposed by Wirges et al. [Wir21]. Initially, a simple pass-through filter is applied to remove points that lie within a certain height range relative to the calibrated ground plane. Next, the surface estimation method is applied to refine the ground surface model, using the filtered points near the ground plane as input. The ground surface is represented by a continuous 2D uniform B-spline surface, providing a smooth and accurate representation of the ground. Points

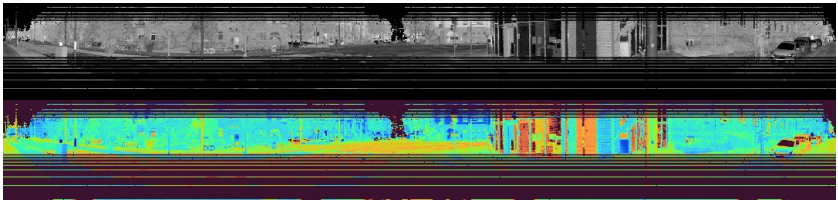
that lie within a specified threshold distance from the estimated ground surface are defined as ground points, while the remaining points are considered non-ground points. In this thesis, the distance threshold is set to 0.2 m.

Once the ground point cloud has been separated, edge and plane features are extracted from the remaining non-ground points. The non-ground points are divided into two separate point clouds: the edge point cloud and the plane point cloud. Subsequently, voxel maps for both the edge and plane features are generated from the respective point clouds.

To separate edge and plane points, the LiDAR sensor model is used to transform the 3D point cloud into a range image: a 2D representation where the x-axis denotes the azimuth angle and the y-axis corresponds to the elevation angle. Examples of the mapped range images are shown in Figure 4.1 for two LiDAR sensors: the Velodyne LiDAR (64 layers) used in the KITTI dataset and the Velodyne Alpha Prime LiDAR (128 layers). Once the range image is generated, the curvature of the points can be computed based on the neighboring points within the image. Using the calculated curvature and heuristic thresholds, edge and plane point clouds are distinguished. This separation technique, introduced by Zhang et al. [Zha14], relies on the principle that if a point's curvature exceeds a predefined threshold, it is classified as an edge point; otherwise, it is classified



(a) Velodyne 64-channel LiDAR sensor.



(b) Velodyne 128-channel LiDAR sensor.

Figure 4.1: Range image examples using two LiDAR sensors. The upper gray image shows the reflected intensities, while the lower color image is provided for better visualization.

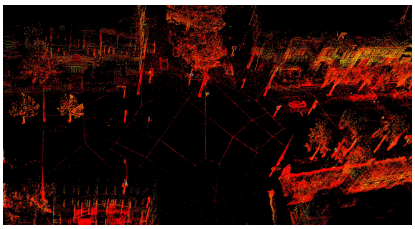
as a plane point. An example of the accumulated edge, plane, and ground point clouds is illustrated in Figure 4.2.

After the edge and plane points are separated, they are mapped into 3D voxels individually. For each voxel, a Gaussian distribution is fitted, and the mean and covariance of the points within that voxel are computed:

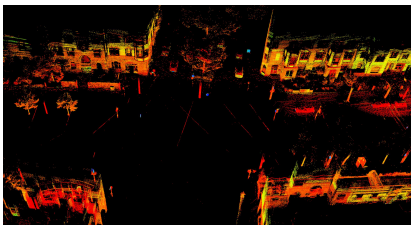
$$\begin{aligned} \mu_i &= \frac{1}{m} \sum_{\mathbf{p}_j \in \mathscr{V}_i} \mathbf{p}_j \\ \Sigma_i &= \frac{1}{m} \sum_{\mathbf{p}_j \in \mathscr{V}_i} (\mathbf{p}_j - \mu_i)(\mathbf{p}_j - \mu_i)^T, \end{aligned} \tag{4.25}$$

where m represents the number of points in the 3D voxel $\mathscr{V}_i$. The mean μ_i and covariance Σ_i are used to classify the type of feature for each 3D voxel. Figure 4.3 demonstrates an example of the LiDAR feature extraction, where the 3D voxels are color-coded to represent edge (yellow) and plane (blue) features.

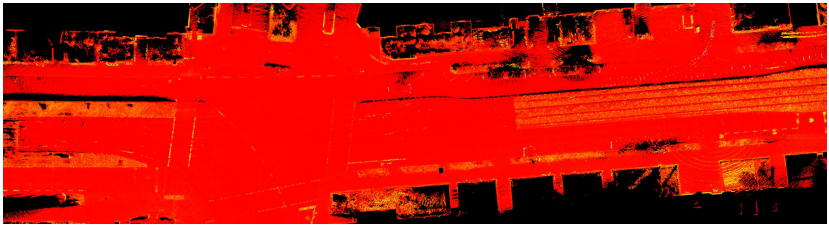
To accumulate the non-ground point clouds, the Normal Distributions Transform (NDT) method is employed to estimate ego-motion. The voxelization



(a) Accumulated edge point cloud.



(b) Accumulated plane point cloud.



(c) Accumulated ground point cloud.

Figure 4.2: Example of accumulated edge, plane, and ground feature point clouds.

process is applied to the accumulated non-ground point clouds, and separate feature maps for edge and plane points are generated. Since the original LiDAR point cloud frame information is preserved, edge and plane feature voxels can be associated across different LiDAR point cloud frames.

Residual Construction and Pose Optimization

Once the voxelization and association processes are complete, edge and plane voxel features are generated, which can then be utilized to construct the bundle adjustment residual functions that are crucial for estimating the LiDAR sensor poses. There are two primary approaches to building these residual functions: the point-to-primitive residual function and the primitive-to-primitive residual function. The following sections discuss the construction of residual functions for a single feature voxel $\mathcal{V}_i$.

Regardless of the type of residual function used, the basic idea is to first select a master frame for the voxel $\mathcal{V}_i$, from which edge or plane features can be fitted using the Gaussian distribution parameters of the LiDAR points. The distance calculation differs depending on whether a point-to-primitive or primitive-to-primitive residual function is applied.

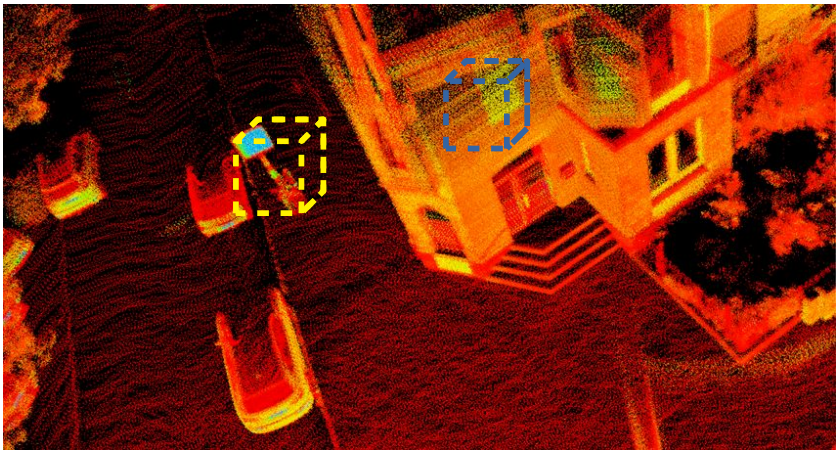


Figure 4.3: LiDAR feature extraction using 3D voxels: the yellow voxel represents an edge feature, while the blue voxel represents a plane feature.

In the point-to-primitive residual function, the distance is computed between LiDAR points from non-master LiDAR point cloud frames and the edge or plane fitted from the master frame. For edge voxel features, the mathematical formulation of the point-to-primitive residual function is expressed as:

$$\mathbf{r}_{\text{point,edge}} = \Omega(\mathbf{n}_{\text{m}}\mathbf{n}_{\text{m}}{}^{\text{T}})(\mathbf{T}^{\text{m}}_{\text{nm}}\mathbf{p}_{\text{nm}} - \mathbf{p}_{\text{m,c}}), \tag{4.26}$$

where $\mathbf{r}$ represents the residual vector, $\mathbf{p}_{\text{nm}}$ is a LiDAR point from a non-master LiDAR point cloud frame, and Ω is the weighting matrix. $\mathbf{T}^{\text{m}}_{\text{nm}}$ is the transformation from the non-master to the master LiDAR point cloud frame, which can be computed by the poses using the following equation:

$$\mathbf{T}^{\text{m}}_{\text{nm}} = \mathbf{T}^{-1}_{\text{m}}\mathbf{T}_{\text{nm}}. \tag{4.27}$$

$\mathbf{p}_{\text{m,c}}$ and $\mathbf{n}_{\text{m}}$ are the center and orientation of the fitted edge in the master frame, respectively. The vector $(\mathbf{T}^{\text{m}}_{\text{nm}}\mathbf{p}_{\text{nm}} - \mathbf{p}_{\text{m,c}})$ represents the displacement from the edge center to the LiDAR point, while $(\mathbf{n}_{\text{m}}\mathbf{n}_{\text{m}}{}^{\text{T}})$ is the covariance matrix of the edge orientation. For plane voxel features, the point-to-primitive residual is formulated as:

$$\mathbf{r}_{\text{point,plane}} = \Omega(\mathbf{I}_{\text{d}} - \mathbf{n}_{\text{m}}\mathbf{n}_{\text{m}}{}^{\text{T}})(\mathbf{T}^{\text{m}}_{\text{nm}}\mathbf{p}_{\text{nm}} - \mathbf{p}_{\text{m,c}}), \tag{4.28}$$

where $\mathbf{I}_{\text{d}}$ is the 3×3 identity matrix.

For the primitive-to-primitive residual error function, an edge or plane is first fitted to LiDAR points from other associated LiDAR point cloud frames, in addition to the master frame. The residual function is then constructed based on the distance between the fitted edge or plane from the master frame and the fitted edge or plane from other associated frames. For edge voxel features, the primitive-to-primitive residual function includes both distance and orientation residuals, which can be formulated as:

$$\begin{aligned} \mathbf{r}_{\text{primitive,edge,dist}} &= \Omega_{\text{dist}}(\mathbf{n}_{\text{m}}\mathbf{n}_{\text{m}}{}^{\text{T}})(\mathbf{T}^{\text{m}}_{\text{nm}}\mathbf{p}_{\text{nm,c}} - \mathbf{p}_{\text{m,c}}) \\ \mathbf{r}_{\text{primitive,edge,orie}} &= \Omega_{\text{orie}}(\mathbf{R}^{\text{m}}_{\text{nm}}\mathbf{n}_{\text{nm}} - \mathbf{n}_{\text{m}}), \end{aligned} \tag{4.29}$$

where $\mathbf{p}_{\text{nm,c}}$ and $\mathbf{n}_{\text{nm}}$ are the center and orientation of the fitted non-master edge, respectively. $\mathbf{R}^{\text{m}}_{\text{nm}}$ is the rotation from the non-master to the master frame, and Ω_{dist} and Ω_{orie} are weighting matrices for distance and orientation residuals.

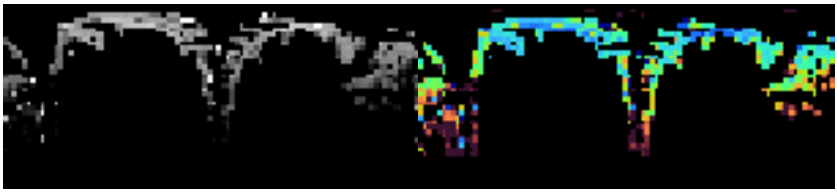
For plane voxel features, the primitive-to-primitive residual function is analogous to that of the edge voxel feature, and it can be expressed as:

$$\begin{aligned} \mathbf{r}_{\text{primitive,plane,dist}} &= \Omega_{\text{dist}}(\mathbf{I}_{\text{d}} - \mathbf{n}_{\text{m}}\mathbf{n}_{\text{m}}{}^{\text{T}})(\mathbf{T}^{\text{m}}_{\text{nm}}\mathbf{p}_{\text{nm,c}} - \mathbf{p}_{\text{m,c}}) \\ \mathbf{r}_{\text{primitive,plane,orie}} &= \Omega_{\text{orie}}(\mathbf{R}^{\text{m}}_{\text{nm}}\mathbf{n}_{\text{nm}} - \mathbf{n}_{\text{m}}), \end{aligned} \tag{4.30}$$

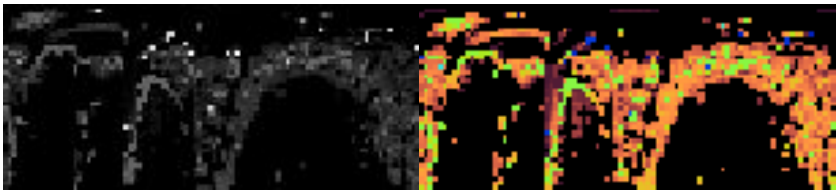
Each method has its advantages and trade-offs. The point-to-primitive residual function generally offers higher accuracy because it utilizes all individual LiDAR points. However, this comes at the cost of higher computational complexity due to the large number of residuals. On the other hand, the primitive-to-primitive residual function has lower computational cost since it operates on fewer residuals. However, their accuracy is compromised because the fitted edge or plane from the non-master LiDAR point cloud frame may not be as precise as that from the master frame due to a smaller number of LiDAR points, and it only considers the distance between fitted primitives. In this thesis, during the offline SLAM process, the point-to-primitive residual function is preferred to achieve better motion estimation accuracy.

4.3 Loop Closure

Loop closure detection plays a critical role in SLAM systems to mitigate pose drift by identifying previously visited locations. This work employs the ISC method [Wan20], which projects LiDAR points onto a 2D polar grid map where the number of points and their aggregated intensities within each grid cell form representative context descriptors. Two examples of ISC mapping are presented in Figure 4.4. Loop closure candidates are detected when the similarity between ISC maps (computed using normalized cross-correlation or cosine similarity) exceeds a predefined threshold. To enhance robustness, only candidates tracked across a specified number of consecutive frames are considered valid.



(a) Velodyne 64-channel LiDAR sensor.



(b) Velodyne 128-channel LiDAR sensor.

Figure 4.4: ISC mapping examples of LiDAR point clouds. The intensity map is presented on the left, while the color visualization is displayed on the right.

4.4 Continuous-Time Trajectory Bundle Adjustment

As motivated in Section 2.3, discrete-time PGO struggles with asynchronous, heterogeneous sensor streams and cannot express velocities or accelerations as continuous quantities. Bundle adjustment over a continuous-time B-spline trajectory addresses these shortcomings: it jointly optimizes all sensor measurements in a unified cost function, incorporates GNSS, IMU, camera, and LiDAR data at their native timestamps without explicit synchronization, and models IMU measurements directly as trajectory derivatives, suppressing integration drift. For these reasons, continuous-time trajectory bundle adjustment is adopted as the core optimization backend of the proposed SLAM system.

In Section 4.1, the continuous-time trajectory is introduced, where position $\mathbf{t}(t) = \{x, y, z\}^{\mathrm{T}}$ and axis-angle representation $\varphi(t) = \{\varphi_1, \varphi_2, \varphi_3\}^{\mathrm{T}}$ are utilized for modeling. For a more compact representation, position and axis-angle are combined into a 6D state vector, denoted as $\mathbf{x}(t) = \{x, y, z, \varphi_1, \varphi_2, \varphi_3\}^{\mathrm{T}}$. The symbol $\mathbf{S}$ is employed to denote the continuous-time trajectory, which consists of two uniform B-splines: $\mathbf{S} = \{\mathbf{S_t}, \mathbf{S_R}\}$, where $\mathbf{S_t}$ represents the translational part and $\mathbf{S_R}$ represents the rotational part.

In Section 4.1.3 and Section 4.1.4, the time derivatives of the uniform B-spline and axis-angle are derived. Using these time derivatives, from the continuous-time trajectory, translation $\mathbf{t}(t)$, rotation $\mathbf{R}(t)$, linear velocity $\mathbf{v}(t)$, linear acceleration $\mathbf{a}(t)$, angular velocity $\omega(t)$, and angular acceleration $\dot{\omega}(t)$ can be efficiently computed from the trajectory $\mathbf{S}(t)$. As part of the estimation process, sensor measurements are also utilized to adjust the trajectory $\mathbf{S}(t)$. This adjustment process is referred to as continuous-time trajectory bundle adjustment.

Sensor measurement models for pose difference measurements, IMU measurements, camera feature-point measurements, LiDAR voxel feature measurements, loop closure measurements, and GNSS measurements are introduced. Based on these models, a cost function incorporating all types of measurements can be constructed, which, when minimized, yields optimal adjustment of the continuous-time trajectory to best satisfy the measured constraints.

4.4.1 Pose Measurement Model

Pose Measurement

Pose $\mathbf{T}(t) = \{\mathbf{t}(t), \mathbf{R}(t)\}$ at a given time t can be estimated as follows:

$$\mathbf{T}(t) = \{\mathbf{t}(t), \mathbf{R}(t)\} = \{\mathbf{S}_{\mathbf{t}}(t), \exp(\mathbf{S}_{\mathbf{R}}(t))\}, \tag{4.31}$$

where the rotation $\mathbf{R}(t)$ is derived from the axis-angle representation $\varphi(t)$ using the exponential map as described in Equation (4.12). To construct the residual function for the pose measurement model, the pose difference between the estimated pose $\hat{\mathbf{T}}(t)$ and the measured pose $\tilde{\mathbf{T}}(t)$ is computed as:

$$\mathbf{r}_{\mathbf{T}} = \Omega(\tilde{\mathbf{T}}(t) \ominus \hat{\mathbf{T}}(t)). \tag{4.32}$$

Here, Ω is the weighting matrix, and $\ominus$ denotes the pose difference operator.

Pose Difference Measurement

To incorporate pose difference measurements from initial odometry or loop closure results, the residual function is formulated as follows:

$$\mathbf{r}_{\Delta\mathbf{T}} = \Omega(\tilde{\mathbf{T}}_{t_1}^{t_2} \ominus \hat{\mathbf{T}}_{t_1}^{t_2}). \tag{4.33}$$

In this case, $\hat{\mathbf{T}}_{t_1}^{t_2}$ represents the pose difference from timestamp t_1 to t_2 as estimated from the continuous-time trajectory, and $\tilde{\mathbf{T}}_{t_1}^{t_2}$ represents the measured pose difference from timestamp t_1 to t_2.

GNSS Global Position Measurement

As a specific case of the pose measurement model, the GNSS measurement model represents the translation component of the pose, where the GNSS global

position measurement $\tilde{\mathbf{t}}(t)$ can be used to define the residual function:

$$\mathbf{r}_{\mathbf{t}} = \Omega(\tilde{\mathbf{t}}(t) - \hat{\mathbf{t}}(t)). \tag{4.34}$$

Here, $\mathbf{t}(t)$ refers to the estimated translation at time t from the continuous-time trajectory $\mathbf{S}(t)$ to be optimized.

Camera Feature Point Measurement

As a result of the image feature matching process outlined in Section 4.2.3, a feature point tracklet $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_i, \ldots, \mathbf{z}_n\}$ is generated, which can be used to formulate the residual function. The image pixel reprojection error, which was discussed in Section 4.2.3 for camera pose estimation, is employed in the camera feature point measurement model to derive the residual function. By combining the reprojection error presented in Equation (4.23) with pose estimation from the continuous-time trajectory $\mathbf{S}(t)$ as given in Equation (4.31), the residual function is expressed as:

$$\mathbf{r}_{\mathcal{Z}} = \Omega(\mathbf{z}_i - \pi_i(\mathbf{K}_i^{-1}\{\mathbf{S}_{\mathbf{t}}(t), \mathbf{S}_{\mathbf{R}}(t)\}_i^{-1}\mathbf{l}_j)). \tag{4.35}$$

In this equation, $\mathbf{z}_i$ represents the image coordinate of the i-th camera feature point, and π_i is the reprojection function based on the camera model. $\mathbf{K}_i^{-1}$ denotes the inverse of the camera intrinsic matrix corresponding to feature point $\mathbf{z}_i$, while $\mathbf{l}_j$ represents the 3D point in the camera coordinate system. The term $\{\mathbf{S}_{\mathbf{t}}(t), \mathbf{S}_{\mathbf{R}}(t)\}^{-1}$ indicates the inverse pose at timestamp t derived from the continuous-time trajectory $\mathbf{S}(t)$, which is subject to optimization.

LiDAR Feature Voxel Measurement

Analogous to the image pixel reprojection error function, LiDAR voxel feature distances, as introduced in Section 4.2.4, are utilized to define the LiDAR measurement residual function. By combining pose estimation from the continuous-time trajectory $\mathbf{S}(t)$, as described in Equation (4.31), with LiDAR

voxel feature distances from Section 4.2.4, the residual function can be formulated. The LiDAR voxel feature distance comprises two types: edge voxel feature distance, as presented in Equation (4.26) or Equation (4.29), and LiDAR plane voxel feature distance, as shown in Equation (4.28) or Equation (4.30). For compactness, $\mathbf{r}_\mathrm{l}$ is used to symbolize the LiDAR voxel feature residual function:

$$\mathbf{r}_{\mathscr{V}} = \Omega(\mathbf{r}_\mathrm{l}(\mathbf{S}(t_\mathrm{nm}), \mathbf{S}(t_\mathrm{m}))), \tag{4.36}$$

$\mathbf{S}(t_\mathrm{m})$ and $\mathbf{S}(t_\mathrm{nm})$ represent the estimated poses from the continuous-time trajectory at timestamp t_m, corresponding to the pose of the master LiDAR point cloud frame, and timestamp t_nm, corresponding to the pose of the non-master LiDAR point cloud frame, respectively. The function $\mathbf{r}_\mathrm{l}(\mathbf{T}_1, \mathbf{T}_2)$ is employed to compute the LiDAR voxel feature residual between poses $\mathbf{T}_1$ and $\mathbf{T}_2$.

4.4.2 Velocity and Acceleration Measurement Model

In Section 4.1.3, the first and second-order time derivatives of the uniform B-spline basis function are presented in Equation (4.13) and Equation (4.14). The state vector for the translation part of the continuous-time trajectory, $\mathbf{S}_\mathbf{t}(t)$, consists of 3D Cartesian coordinates $\mathbf{t}(t) = \{x, y, z\}^\mathrm{T}$, and linear velocity $\mathbf{v}(t)$ and linear acceleration $\mathbf{a}(t)$ can be readily estimated from the continuous-time trajectory $\mathbf{S}(t)$ using the following estimator:

$$\begin{aligned}
\mathbf{v}(t) &= \frac{\mathrm{d}\mathbf{t}(t)}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{S}_\mathbf{t}(t)}{\mathrm{d}t} \\
&= \frac{\mathrm{d}\sum_{i=0}^{n} B_{\mathrm{i,k}}(t)\mathbf{p}_\mathrm{i}^\mathrm{c}}{\mathrm{d}t}, \\
&= \sum_{i=0}^{n} \frac{\mathrm{d}B_{\mathrm{i,k}}(t)}{\mathrm{d}t}\mathbf{p}_\mathrm{i}^\mathrm{c}
\end{aligned} \tag{4.37}$$

where $\mathbf{p}_\mathrm{i}^\mathrm{c}$ represents the 3D translation control point of the uniform B-spline $\mathbf{S}_\mathbf{t}(t)$, and $\frac{\mathrm{d}B_{\mathrm{i,k}}(t)}{\mathrm{d}t}$ denotes the first-order time derivative of the uniform B-spline basis function, which can be calculated using Equation (4.13). Similarly, linear

acceleration $\mathbf{a}(t)$ can be estimated using:

$$\begin{aligned}\mathbf{a}(t) &= \frac{\mathrm{d}^2\mathbf{t}(t)}{\mathrm{d}t^2} = \frac{\mathrm{d}^2\mathbf{S}_{\mathbf{t}}(t)}{\mathrm{d}t^2} \\ &= \frac{\mathrm{d}^2 \sum_{i=0}^{n} B_{\mathrm{i,k}}(t)\mathbf{p}_{\mathrm{i}}^{\mathrm{c}}}{\mathrm{d}t^2}. \\ &= \sum_{i=0}^{n} \frac{\mathrm{d}^2 B_{\mathrm{i,k}}(t)}{\mathrm{d}t^2}\mathbf{p}_{\mathrm{i}}^{\mathrm{c}}\end{aligned} \tag{4.38}$$

where $\frac{\mathrm{d}^2 B_{\mathrm{i,k}}(t)}{\mathrm{d}t^2}$ is the second-order time derivative of the uniform B-spline basis function, which can be calculated using Equation (4.14).

To estimate angular velocity $\omega(t)$ and angular acceleration $\dot{\omega}(t)$ from the continuous-time rotation trajectory $\mathbf{S}_{\mathbf{R}}(t)$, the time derivative of the uniform B-spline $\mathbf{S}_{\mathbf{R}}(t)$ must first be computed. However, unlike the intuitive calculation of linear velocity $\mathbf{v}(t)$ and linear acceleration $\mathbf{a}(t)$ from the translation vector $\mathbf{t}(t)$, angular velocity $\omega(t)$ and angular acceleration $\dot{\omega}(t)$ cannot be directly computed from the axis-angle $\varphi(t)$, but rather from the rotation matrix $\mathbf{R}(t)$. The derivative of the 3D rotation matrix $\mathbf{R}(t)$ with respect to the axis-angle parameterization $\varphi(t)$ is introduced in Equation (4.15) or Equation (4.16) when $\varphi(t) \to \mathbf{0}$. By combining this with the calculation of angular velocity $\omega(t)$ and angular acceleration $\dot{\omega}(t)$ from the rotation matrix $\mathbf{R}(t)$, as presented in Equation (4.19) and Equation (4.20), an estimator for these quantities can be constructed. Angular velocity $\omega(t)$ can be estimated from the continuous-time rotation trajectory $\mathbf{S}_{\mathbf{R}}(t)$ using:

$$\begin{aligned}[\omega(t)]_{\times} &= \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\mathbf{R}^{\mathrm{T}} = \left[\frac{\partial\mathbf{R}}{\partial\varphi_1}, \frac{\partial\mathbf{R}}{\partial\varphi_2}, \frac{\partial\mathbf{R}}{\partial\varphi_3}\right] \frac{\mathrm{d}\varphi(t)}{\mathrm{d}t}\mathbf{R}^{\mathrm{T}} \\ &= \left[\frac{\partial\mathbf{R}}{\partial\varphi_1}, \frac{\partial\mathbf{R}}{\partial\varphi_2}, \frac{\partial\mathbf{R}}{\partial\varphi_3}\right] \frac{\mathrm{d}\mathbf{S}_{\mathbf{R}}(t)}{\mathrm{d}t}\mathbf{R}^{\mathrm{T}} \\ &= \left[\frac{\partial\mathbf{R}}{\partial\varphi_1}, \frac{\partial\mathbf{R}}{\partial\varphi_2}, \frac{\partial\mathbf{R}}{\partial\varphi_3}\right] \left(\sum_{i=0}^{n} \frac{\mathrm{d}B_{\mathrm{i,k}}(t)}{\mathrm{d}t}\mathbf{p}_{\mathrm{i}}^{\mathrm{c}}\right)\mathbf{R}^{\mathrm{T}}\end{aligned} \tag{4.39}$$

From the skew-symmetric matrix $[\omega(t)]_\times$, the angular velocity $\omega(t)$ can be easily derived. Similarly, to estimate angular acceleration $\dot{\omega}(t)$, Equation (4.20) can be applied, yielding:

$$\begin{aligned}[\dot{\omega}(t)]_\times &= \frac{\mathrm{d}^2\mathbf{R}}{\mathrm{d}t^2}\mathbf{R}^\mathrm{T} + \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\left(\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t}\right)^\mathrm{T} \\ &= \left[\frac{\partial\mathbf{R}}{\partial\varphi_1}, \frac{\partial\mathbf{R}}{\partial\varphi_2}, \frac{\partial\mathbf{R}}{\partial\varphi_3}\right]\frac{\mathrm{d}^2\mathbf{S}_\mathbf{R}(t)}{\mathrm{d}t^2}\mathbf{R}^\mathrm{T} \\ &+ \left[\frac{\partial\mathbf{R}}{\partial\varphi_1}, \frac{\partial\mathbf{R}}{\partial\varphi_2}, \frac{\partial\mathbf{R}}{\partial\varphi_3}\right]\frac{\mathrm{d}\mathbf{S}_\mathbf{R}(t)}{\mathrm{d}t}\left(\left[\frac{\partial\mathbf{R}}{\partial\varphi_1}, \frac{\partial\mathbf{R}}{\partial\varphi_2}, \frac{\partial\mathbf{R}}{\partial\varphi_3}\right]\frac{\mathrm{d}\mathbf{S}_\mathbf{R}(t)}{\mathrm{d}t}\right)^\mathrm{T}\end{aligned}, \tag{4.40}$$

where $\frac{\mathrm{d}^2\mathbf{S}_\mathbf{R}(t)}{\mathrm{d}t^2}$ and $\frac{\mathrm{d}\mathbf{S}_\mathbf{R}(t)}{\mathrm{d}t}$ can be computed using Equation (4.14) and Equation (4.13). The matrix $[\dot{\omega}(t)]_\times$ is the skew-symmetric matrix of the angular acceleration $\dot{\omega}(t)$, and the angular acceleration can be easily derived from it.

IMU Measurement

The measurement model for linear acceleration $\mathbf{a}(t)$ is derived in Equation (4.38), and the corresponding residual function can be formulated as:

$$\mathbf{r}_\mathbf{a} = \Omega(\tilde{\mathbf{a}}(t) - \hat{\mathbf{a}}(t)). \tag{4.41}$$

where $\hat{\mathbf{a}}(t)$ represents the optimized linear acceleration at timestamp t from the rotation part of the continuous-time trajectory $\hat{\mathbf{S}}_\mathbf{R}(t)$, and $\tilde{\mathbf{a}}(t)$ denotes the linear acceleration measured by the IMU sensor at the same timestamp t. Similarly, the residual function for angular velocity $\omega(t)$ can be formulated as:

$$\mathbf{r}_\omega = \Omega(\tilde{\omega}(t) - \hat{\omega}(t)), \tag{4.42}$$

where $\hat{\omega}(t)$ denotes the optimized angular velocity at timestamp t from the rotation part of the continuous-time trajectory $\hat{\mathbf{S}}_\mathbf{R}(t)$, and $\tilde{\omega}(t)$ is the angular velocity measured by the IMU sensor at timestamp t.

5 Semantic Parametric Mapping

As discussed in Section 2.4, there is a growing body of research dedicated to semantic mapping of 3D environments for autonomous driving. A highly accurate and robust representation of 3D driving environments can extend the perception range while providing valuable contextual information for decision-making and trajectory planning. Semantic primitives, such as poles, traffic lights, traffic signs, and road markings, exhibit long-term stability, making them well-suited for map representation. To optimize storage efficiency, these semantic primitives should be encoded using parametric models, including attributes such as position, orientation, and shape.

This chapter introduces the semantic parametric mapping system, which integrates multi-modal sensor data, including camera images, LiDAR point clouds, semantic/instance segmentation masks, and ego-motion, to generate a structured semantic parametric primitive map. Semantic primitives in the mapping framework are categorized into two primary types:

- **Elevated Primitives**: poles, traffic lights, and traffic signs.
- **Road-Marking Primitives**: dashed lines, arrows, solid lines, and curbs.

The overall architecture of the semantic parametric mapping system is depicted in Figure 5.1. Ground surface estimation and road image pixel reconstruction are detailed in Section 5.1. The detection of elevated and road-marking primitives is presented in Section 5.2. The methodology for associating detected primitive instances is introduced in Section 5.3. Parametric modeling and optimization for elevated and road-marking primitives are discussed in Section 5.4 and Section 5.5, respectively. Finally, the generated exemplary semantic primitive maps are visualized in 3D space in Section 5.6.

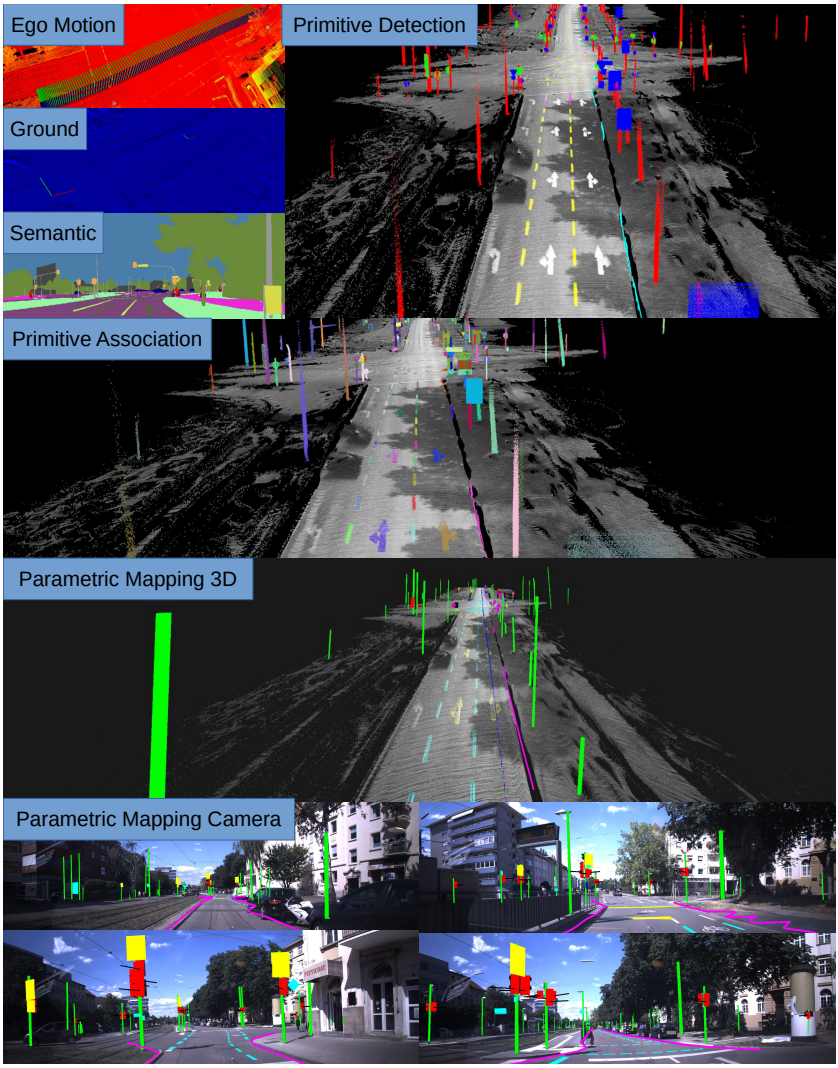


Figure 5.1: The architecture of the semantic parametric mapping system for elevated primitives, including poles, traffic lights, and traffic signs, as well as road-marking primitives, such as dashed lines, arrows, solid lines, and curbs.

5.1 Ground Surface Estimation

For the detection of road marking primitives, an accurate representation of ground surfaces is essential. To achieve this, ground surfaces are modeled using uniform B-spline surfaces. The parameters of the uniform B-spline surface are optimized based on accumulated LiDAR ground point clouds over time, which serve as measurements. The estimated uniform B-spline surface provides a mapping function from the Cartesian coordinates (x, y) to the height z:

$$z = \mathbf{S}_{\mathrm{g}}(x, y), \tag{5.1}$$

where $\mathbf{S}_{\mathrm{g}}$ represents the uniform B-spline surface model of the ground surface. An example of an estimated ground surface uniform B-spline model is illustrated in Figure 5.2 using a 3D mesh representation.

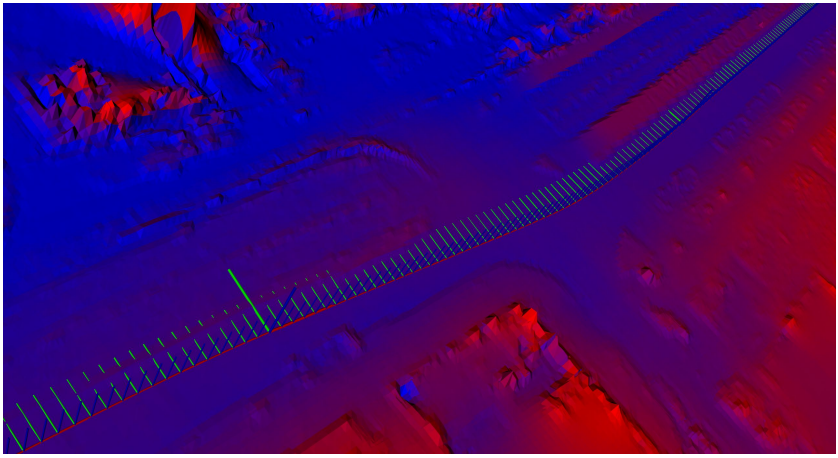


Figure 5.2: Example of an estimated ground surface. The mesh colors indicate height variation (blue → low, red → high), while the small coordinate frames represent the motion trajectory of the ego-vehicle during data recording, and the large coordinate frame denotes the coordinate system origin of the ground surface.

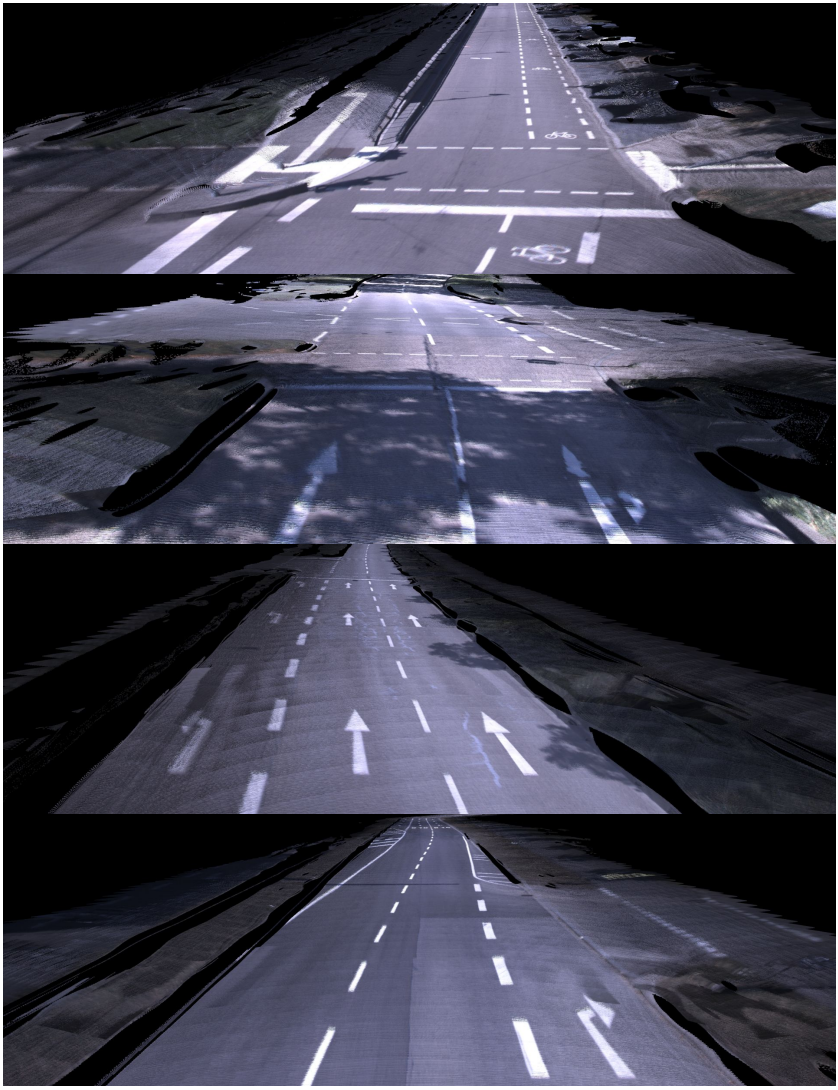


Figure 5.3: Examples of image pixel-based road surface reconstruction. Image pixels are projected onto the ground surface, generating a corresponding 3D point cloud.

5.1.1 Image Pixel Reconstruction

To project image pixels onto the estimated ground surface, the camera intrinsics introduced in Section 3.1 are utilized to generate a 3D viewing ray ρ for each image pixel (u, v):

$$\rho = \{\rho_s, \rho_d\} = \mathbf{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \tag{5.2}$$

where ρ represents the 3D viewing ray, consisting of the support point ρ_s and the 3D direction vector ρ_d. Here, $\mathbf{K}$ denotes the camera intrinsic matrix, while (u, v) are the image pixel coordinates. Subsequently, the intersection point between the 3D viewing ray and the estimated ground surface is computed. A 3D point $\mathbf{p}(d)$ along the viewing ray ρ can be expressed as:

$$\mathbf{p}(d) = \rho_s + d\rho_d, \tag{5.3}$$

where $d \in \mathbb{R}^+$ represents the depth parameter along the viewing ray, measuring the distance from the support point ρ_s to the 3D point $\mathbf{p}(d)$. To determine the intersection point between the 3D viewing ray and the ground surface, the optimal depth parameter $\hat{d}$ is estimated iteratively by minimizing:

$$\hat{d} = \arg\min_d \left\| \mathbf{S}_g(x, y) - (\rho_s + d\rho_d)_z \right\|^2. \tag{5.4}$$

Here, $(\cdot)_z$ extracts the z-component of a 3D point. For the iterative refinement of the depth parameter d, an initial starting point d_0 is selected based on the assumption that the ground surface corresponds to a calibrated reference plane positioned at a predefined distance beneath the camera sensor. The optimal depth parameter $\hat{d}$ is then iteratively refined through minimization of the residual between the estimated 3D point's z-component and the corresponding ground surface height at the projected (x, y) coordinates. As a result of the image pixel reconstruction process, dense image pixel-based 3D point clouds are generated. Several examples of these reconstructed 3D point clouds from various driving scenarios are visualized in Figure 5.3.

5.2 Primitive Detection

During the primitive detection process, both elevated and road-marking primitives are identified as distinct instances for each sensor data frame. For each detected primitive instance, a corresponding instance point cloud and an instance semantic mask are generated.

5.2.1 Elevated Primitives

To detect elevated primitives such as poles, traffic lights, and traffic signs with various shapes, instance-segmented masks from camera images are utilized. Detection is performed using the Seamless Segmentation approach proposed by Porzi et al. [Por19]. As an outcome of Seamless Segmentation, 2D bounding boxes and semantic masks are generated for each segmented instance across the entire image. An example result is shown in Figure 5.4, where the combined mask for all detected instances is visualized. To obtain depth information for these instances, the non-ground LiDAR point cloud, generated using the approach outlined in Section 4.2.4, is employed.

The non-ground LiDAR point cloud is projected onto the camera image using the LiDAR-to-camera calibration parameters and the camera model described in Section 3.1. During this projection process, a camera projection map Φ_{map} is generated, containing both depth and reflectivity information. To account for the parallax effect caused by the different mounting positions of the LiDAR and

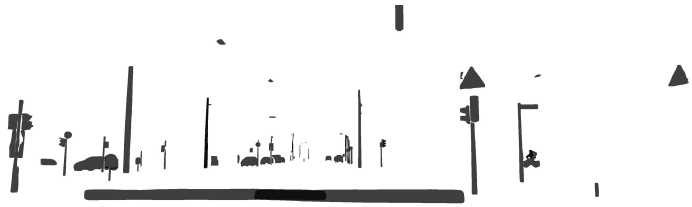


Figure 5.4: Example result of Seamless Segmentation visualized with the combined mask.

camera sensors, the approach introduced by Pauls [Pau25] is applied. Once the camera projection map Φ_{map} is constructed, the instance point cloud is extracted by selecting points from the projection map Φ_{map} that fall within the instance segmentation masks. An example visualization of the camera projection map Φ_{map} is provided in Figure 5.5.

Despite compensating for the parallax effect, detection results may still contain inaccuracies, which could hinder the association of primitives and the estimation of model parameters. To refine the detection results, the instance point cloud undergoes further clustering using the Density-Based Spacial Clustering of Applications with Noise (DBSCAN) algorithm Ester et al. [Est96]. The Euclidean distance between points is utilized as the clustering metric. To eliminate noise, a minimum cluster size of 4 points is enforced. Among the detected clusters, the one closest to the camera sensor is selected as the refined instance point cloud. An example of the detected elevated primitives from a single sensor data frame is illustrated in Figure 5.6.

5.2.2 Road-Marking Primitives

Road marking primitives include dashed lines, arrows, solid lines, and curbs. The detection of road marking primitives is based on semantic segmentation results from camera images and the estimated ground surface. To obtain semantic segmentation results, a custom-trained semantic segmentation network

Figure 5.5: Camera projection map Φ_{map} used for parallax correction in elevated primitive detection. The camera projection map Φ_{map} stores the distance of objects to the camera sensor. Color brightness represents intensity values.

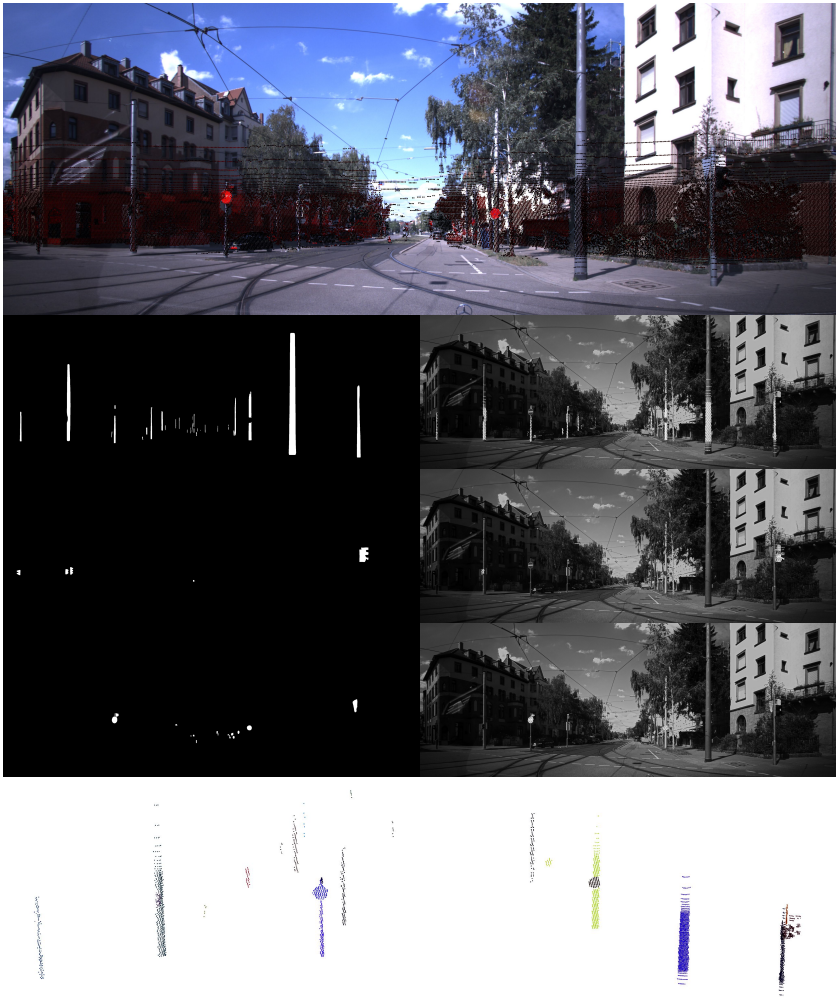


Figure 5.6: Example of elevated primitive detection for a single sensor data frame. From top to bottom, the images illustrate: the camera image with the projected LiDAR point cloud (red points), the instance mask (left) and instance point cloud (right) for poles, the instance mask (left) and instance point cloud (right) for traffic lights, the instance mask (left) and instance point cloud (right) for traffic signs, and the fused instance point cloud of all detections, with different instances highlighted in distinct colors.

proposed by Bieder et al. [Bie23] is utilized. This network is trained on automatically generated real-world data, which is derived from a manually annotated 3D HD semantic map and a high-precision 6D multi-session camera localization approach presented by Sons [Son21]. Dynamic objects in the training data are automatically masked using segmentation masks generated by the Nvidia Semantic Segmentation network Tao et al. [Tao20]. The resulting semantic masks classify the following categories: road, dashed line, solid line, other line, walkway, terrain, dynamic object, and background. Examples of semantic segmentation results are illustrated in Figure 5.7.

Since road marking primitives are located on the ground surface, and this surface is typically confined to the lower portion of camera images, only a cropped

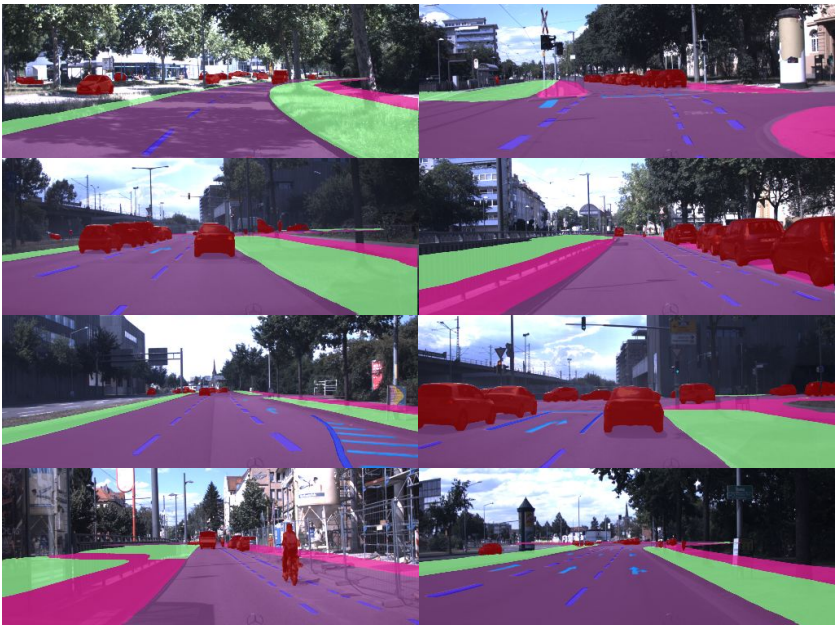


Figure 5.7: Examples of semantic segmentation results overlaid on a camera image. The semantic mask includes the following classes: road (purple), dashed lines (blue), solid lines (heavy blue), other lines (light blue), walkway (purple), terrain (green), dynamic objects (red), and background (no color). It is applied to the road-relevant region, which is a cropped portion of the lower-middle section of the camera image.

rectangular region of the original camera image is used for semantic segmentation. Ground surfaces are estimated by processing LiDAR ground points obtained through the approach described in Section 4.2.4 and the ground surface estimation method introduced in Section 5.1.

As outlined in Section 5.1.1, the intersection point of a 3D viewing ray ρ generated from an image pixel (u, v) and the corresponding ground surface $\mathbf{S}_{\mathrm{g}}$ is computed. Unlike the detection of elevated primitives, where non-ground LiDAR point clouds are projected onto the camera image plane, road marking primitives are detected by projecting semantic mask pixels directly onto the estimated ground surface to generate instance point clouds. This method enables the generation of denser instance point clouds for road marking primitives, as it directly utilizes image pixels rather than relying on LiDAR point clouds. However, the projection accuracy is highly dependent on the distance between the road marking primitives and the camera sensor. In this thesis, only road marking instances within a 15-meter range from the camera sensor are considered for each sensor data frame. An example of the detected road marking primitives for a single sensor data frame is visualized in Figure 5.8.

5.2.3 Instance Map

Following the detection of elevated and road marking primitives, a global instance map is constructed by integrating detected instances from a sequence of sensor data frames, leveraging the estimated ego-motion trajectory for spatial alignment. Illustrative examples of instance maps generated in three different driving scenarios are presented in Figure 5.9.

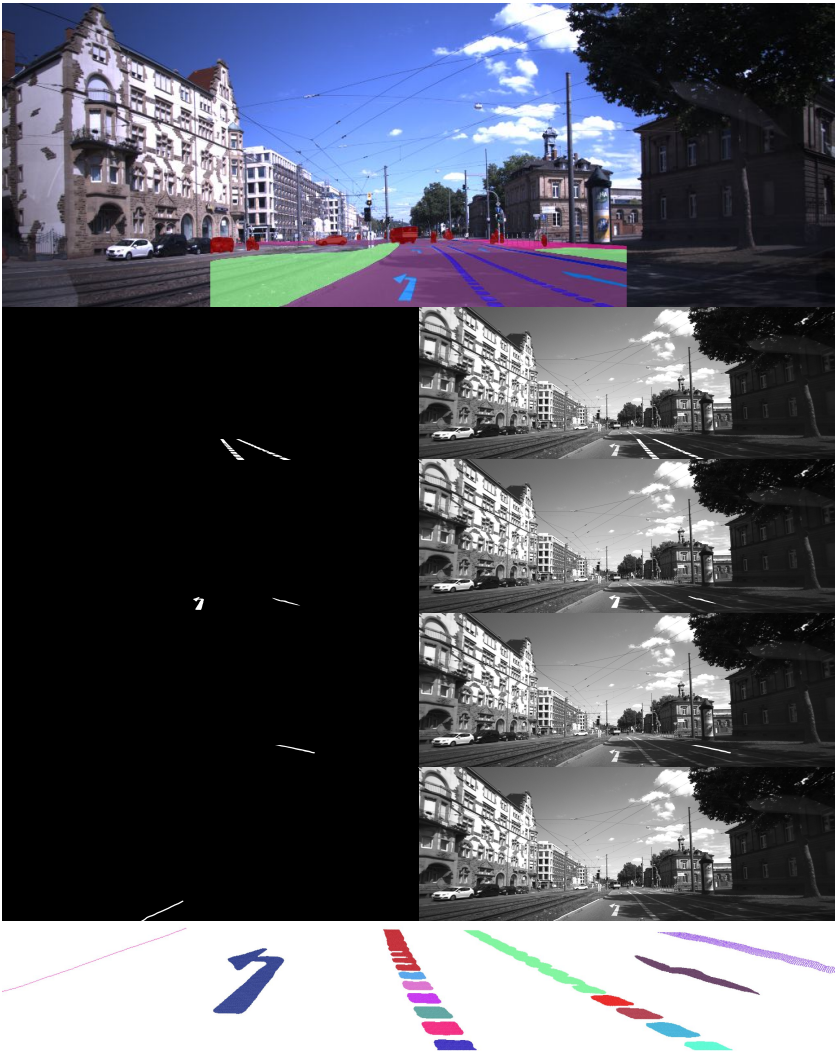


Figure 5.8: Example of road marking primitive detection for a single sensor data frame. From top to bottom, the images illustrate: the camera image with an overlaid semantic segmentation mask, detected dashed lines, arrows, solid lines, and curbs, and the fused instance point cloud, where different colors represent different instances.

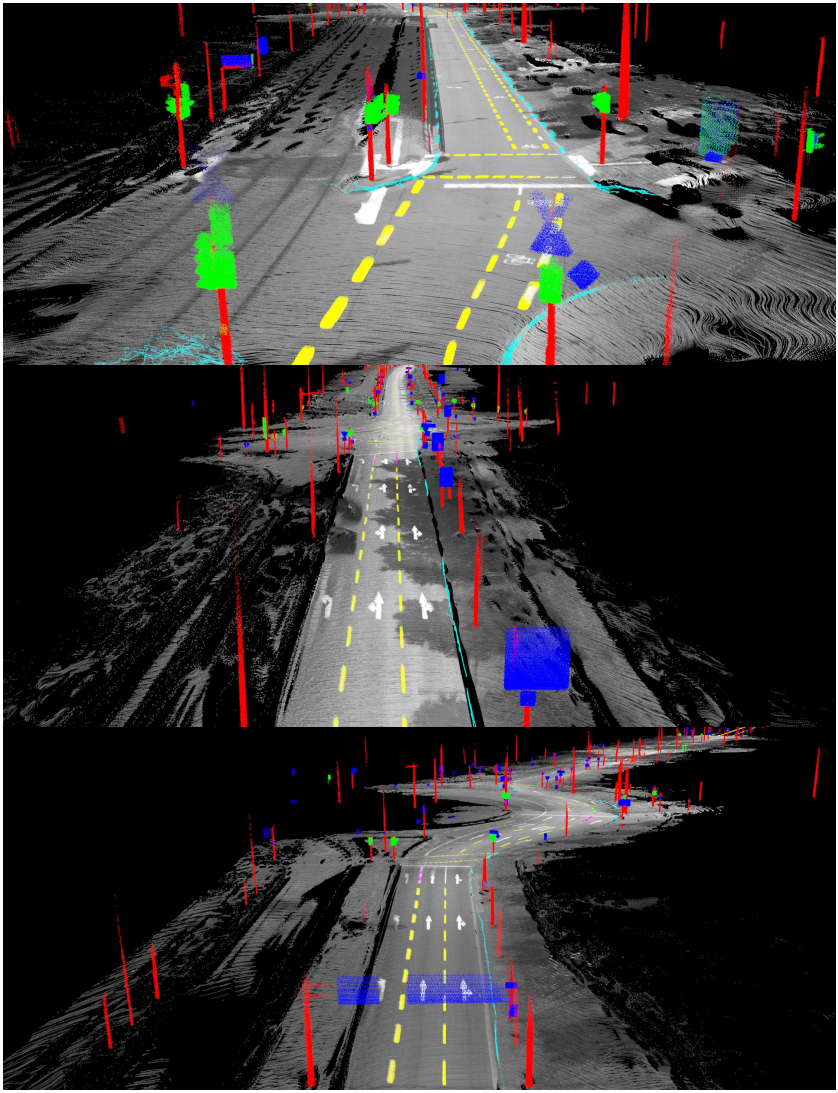


Figure 5.9: Exemplary instance maps from three driving scenarios, illustrating detected elevated and road marking primitives over multiple sensor data frames. Different classes are color-coded: poles (red), traffic lights (green), traffic signs (blue), dashed lines (yellow), arrows (white), solid lines (purple), and curbs (light blue).

5.3 Primitive Association

As introduced in Section 5.2, elevated and road marking primitives are detected as discrete instances. Each instance consists of a semantic mask and a corresponding point cloud, referred to as the instance point cloud in subsequent discussions. To construct a coherent representation of the environment, these detected primitive instances must be associated over time within a global map frame using the estimated ego-motion trajectory. The association process is performed incrementally, frame by frame, by employing a nearest neighbor search followed by a geometric consistency validation.

Initially, a landmark list is initialized using the detected instances from the first sensor data frame. Subsequent instances from later frames are then iteratively associated with the existing landmarks through nearest neighbor search, supplemented by a geometric consistency check, which will be detailed in the following sections. If an instance does not correspond to any existing landmark, a new landmark is instantiated using the newly detected instance. This process is repeated for all sensor data frames. The final output of the association process is a list of landmarks, each consisting of an accumulated landmark point cloud and a corresponding set of associated instances. This landmark information serves as the foundation for subsequent primitive parameter optimization.

Since the quality of primitive detection is generally higher for instances that appear at a closer distance to the camera sensor, and the accuracy of landmark initialization significantly affects the association process, association is performed in reverse chronological order to mitigate the influence of poor detections at larger distances. This approach ensures that instances detected with higher accuracy at closer distances are associated first.

The association process is categorized into two distinct cases: continuous primitives and isolated primitives. Continuous primitives, such as solid lines and curbs, typically extend beyond the field of view of a single sensor frame and are only partially observed in each frame. In contrast, isolated primitives, including poles, traffic lights, traffic signs, dashed lines, and arrows, are generally captured in their entirety within a single sensor frame.

5.3.1 Continuous Primitives

The association of continuous primitive instances is performed using a clustering-based approach. A detected continuous primitive instance is assigned to an existing landmark if the minimum Euclidean distance between any point in the detected instance point cloud and any point in the landmark point cloud falls below a predefined threshold, set to 1.5 *m* in this thesis. If no existing landmark satisfies this condition, a new landmark is initialized using the detected primitive instance. Figure 5.10 illustrates two example results of continuous primitive associations.

5.3.2 Isolated Primitives

The association of isolated primitives is performed using a combination of nearest neighbor search and geometric consistency validation. For nearest neighbor

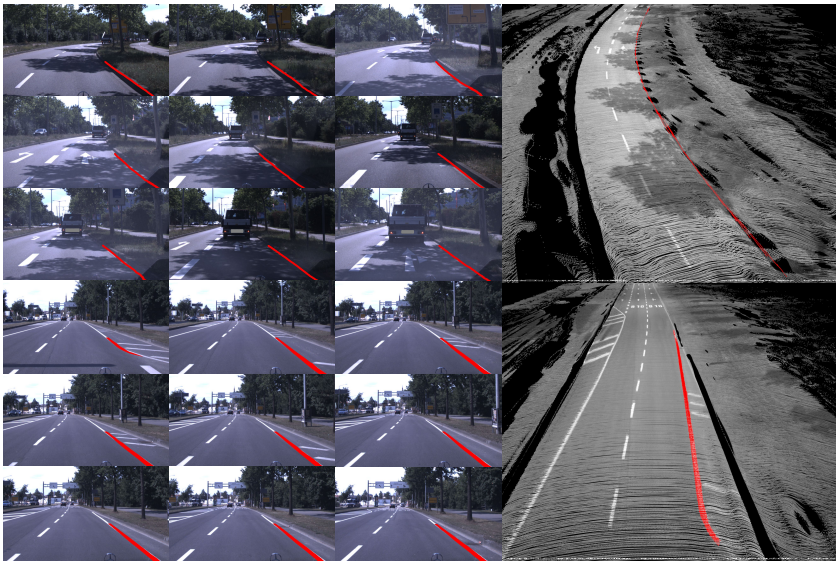


Figure 5.10: Exemplary association results for continuous primitives: solid line (bottom) and curb (top). For each landmark, the key-frame instance semantic masks are displayed on the left, while the accumulated point cloud is shown on the right.

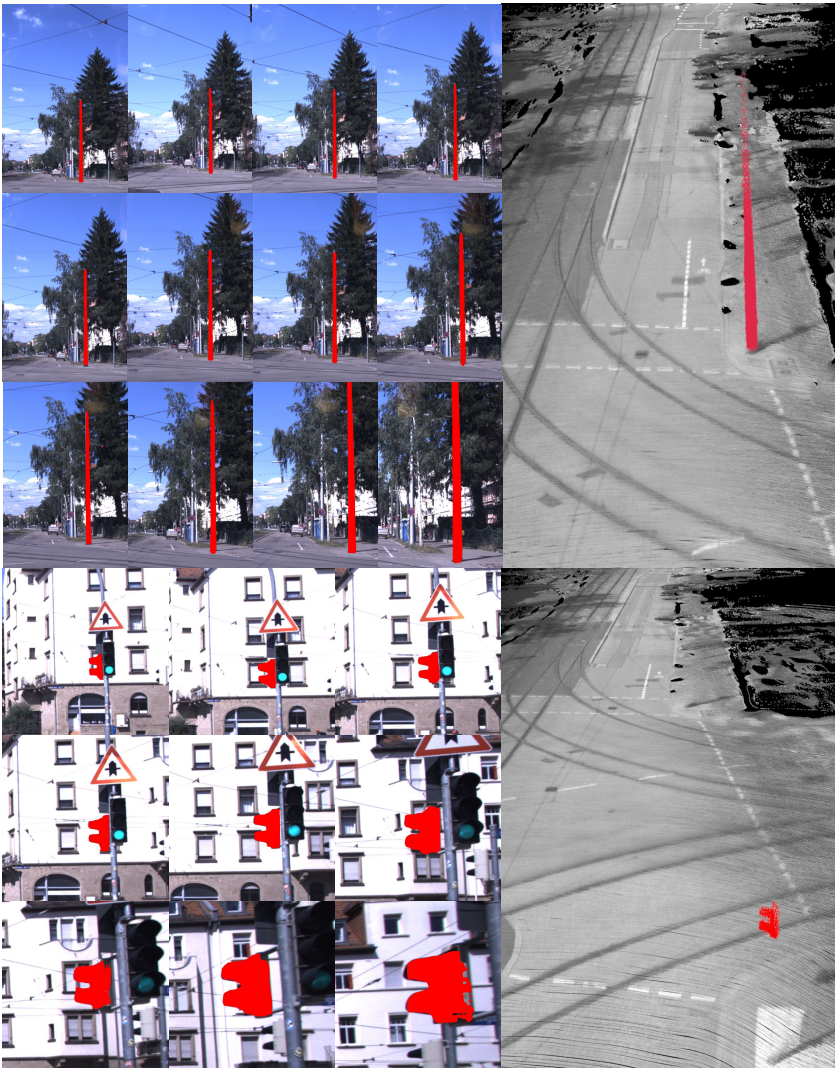


Figure 5.11: Exemplary association results for isolated elevated primitives: pole (top) and traffic light (bottom). For each landmark, the key-frame instance masks are displayed on the left, and the accumulated landmark point cloud is shown on the right.

search, kd-trees are constructed using the center points of existing landmarks, grouped by primitive type. The center points of detected instances are then

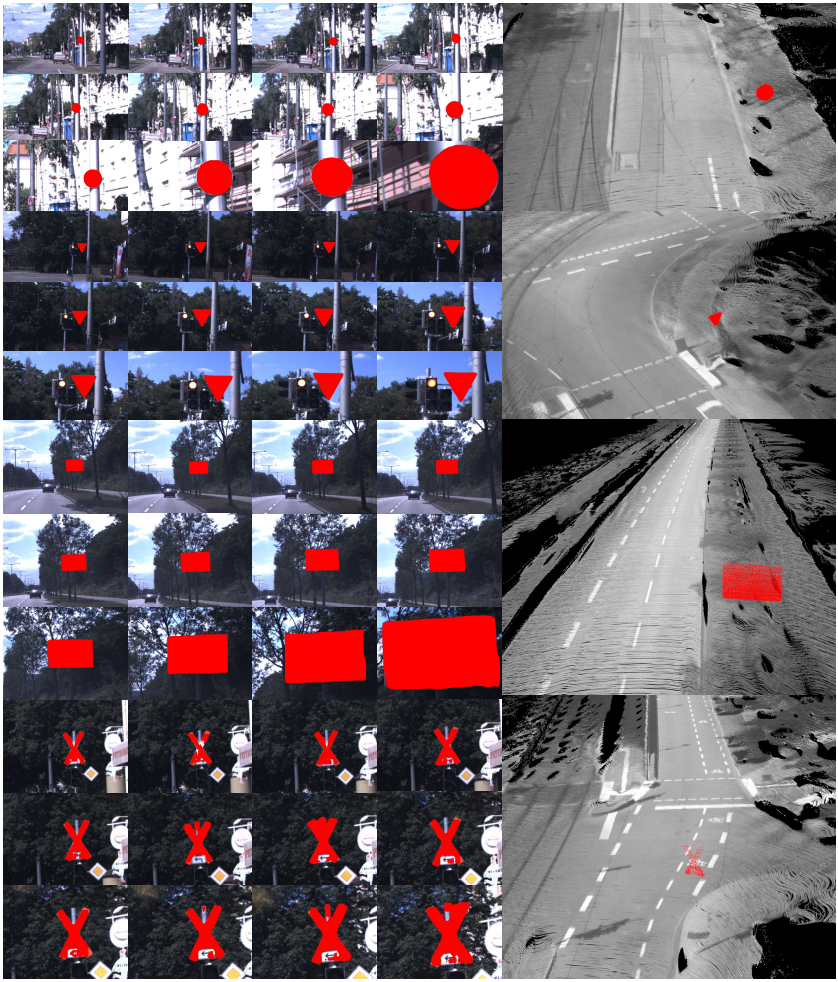


Figure 5.12: Exemplary association results for isolated elevated primitives: traffic signs with different shapes, circle, triangle, rectangle, and other, from top to bottom. For each landmark, the key-frame instance semantic masks are shown on the left, while the accumulated landmark point cloud is shown on the right.

queried in the corresponding kd-tree. If the Euclidean distance between a detected instance and an existing landmark is below a predefined threshold, the instance is initially assigned to the landmark. The threshold values used in this thesis are 0.5 *m* for poles, traffic lights, dashed lines, and arrows, and 1.5 *m* for traffic signs. A larger threshold is used for traffic signs because they vary in size and the estimated center points may deviate due to partial detections.

To further improve association accuracy, a geometric consistency check is performed. The landmark center is projected into the camera image using the ego-motion estimate and camera model to verify whether the projected point falls inside the corresponding semantic mask of the detected instance. The ratio of points within the semantic mask relative to the total tracked instances for the landmark is computed as the inside ratio. If this ratio exceeds a predefined

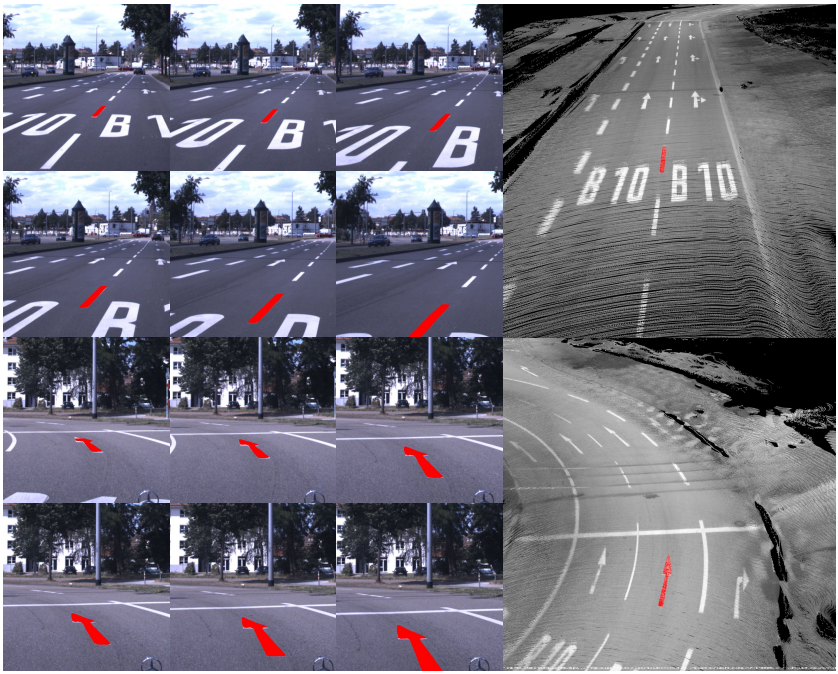


Figure 5.13: Exemplary association results for isolated road marking primitives: dashed line (top) and arrow (bottom). For each landmark, the key-frame instance masks are displayed on the left, and the accumulated landmark point cloud is shown on the right.

threshold, the detected instance is associated with the landmark. Otherwise, a new landmark is initialized. In this thesis, the inside ratio threshold is set to 0.2.

Figure 5.11 and Figure 5.12 illustrate six example association results for isolated elevated primitives including poles, traffic lights, and traffic signs with different shapes. Figure 5.13 presents two examples of association results for isolated road marking primitives, including dashed lines and arrows.

5.3.3 Landmark Map

Upon completion of the association process, a set of primitive landmarks is established. Each landmark comprises an accumulated landmark point cloud and a set of associated primitive instance masks. Additionally, the poses of the associated instances are retained for subsequent model parameter optimization. Due to computational memory constraints, the association process is conducted within localized sub-areas, each containing a limited sequence of sensor data frames. The resulting maps within these sub-areas are referred to as submaps in this thesis. Figure 5.14 presents examples of estimated landmark maps from three distinct driving scenarios.

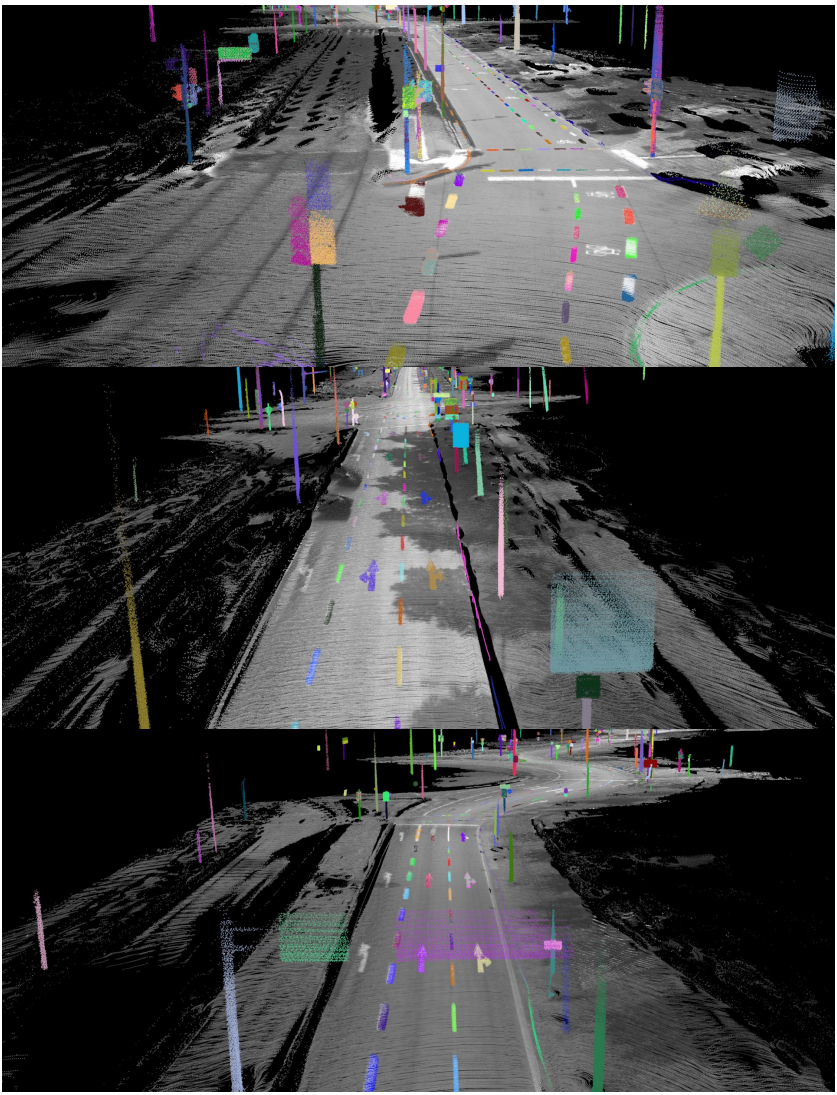


Figure 5.14: Exemplary landmark maps visualizing associated primitive instances from three driving scenarios. Different landmarks are represented with distinct colors. Instances with overlapping colors appear due to the merging of adjacent submaps.

5.4 Parametric Mapping of Elevated Primitives

Following the primitive association process, the next step in processing elevated primitives involves parametric modeling and model parameter optimization. Appropriate geometric models are formulated for various types of elevated primitives. The parameters of these models are initially estimated from the accumulated landmark point cloud and subsequently refined using the corresponding instance point clouds and semantic masks.

To optimize the model parameters, multiple residual functions are formulated, and a non-linear optimization problem is constructed. Upon completion of the optimization process, landmarks are represented as parametric geometric models rather than raw point clouds and semantic masks. The advantages of employing parametric models over point clouds and semantic masks can be summarized as follows:

- **Compact representation**: Significantly reduces storage requirements.
- **Enhanced informativeness**: Facilitates further processing and higher-level reasoning.
- **Improved accuracy and robustness**: Aggregating instance detections across multiple frames minimizes mapping errors.

5.4.1 Pole

Parametric Modeling

As demonstrated in the primitive association results in Figure 5.11, poles are typically observed as vertical cylindrical structures in 3D space. Consequently, they can be effectively modeled using vertical cylinders. However, real-world poles are not always perfectly vertical to the ground surface and may exhibit varying inclinations. Additionally, poles do not always maintain a uniform diameter along their height; they often exhibit a tapering effect, with a larger diameter at the base that gradually decreases towards the top. To better capture

these variations, both a cylindrical and a conical model with a pitched center line axis are introduced.

The parametric modeling approaches for cylindrical and conical pole representations are illustrated in Figure 5.15. The cylindrical pole model is characterized by nine parameters: the center line axis $\mathbf{n} = (n_x, n_y, n_z)^T$, the central reference point $\mathbf{o} = (x_o, y_o, z_o)^T$, the bottom and top distances from the reference point along the center line h_{min} and h_{max}, respectively, and a constant radius r. For any point $\mathbf{p} = (x, y, z)^T$ on the surface of the cylindrical model, the following relationship holds:

$$\frac{\|(\mathbf{p} - \mathbf{o}) \times \mathbf{n}\|}{\|\mathbf{n}\|} = r, \tag{5.5}$$

where $\times$ denotes the cross product and $\|\cdot\|$ represents the vector norm. This equation ensures that every surface point $\mathbf{p}$ maintains a fixed radial distance r from the center line axis $\mathbf{n}$.

For the conical pole model, the parameters remain the same as those of the cylindrical model, except for the radius r. Instead of being constant, the radius $r(h)$ varies as a linear function of the relative height h with respect to the center point $\mathbf{o}$. This variation is governed by two parameters: the base radius at the center height r_o and the diameter variation rate s along the center line $\mathbf{n}$. The

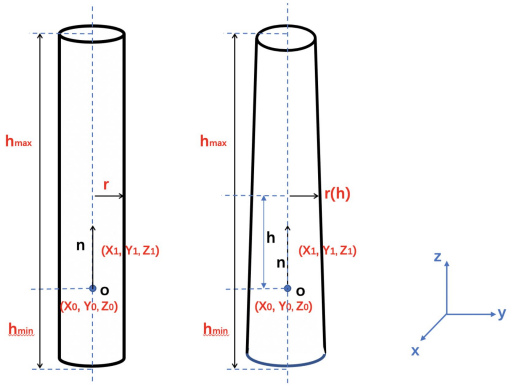


Figure 5.15: Parametric modeling of poles in 3D space: the left model represents a cylinder, while the right model represents a cone.

radius as a function of height is defined as:

$$r(h) = r_{\mathrm{o}} + s \cdot h. \tag{5.6}$$

Similar to the cylindrical model, the surface points of the conical model satisfy the following relationship:

$$\frac{\|(\mathbf{p} - \mathbf{o}) \times \mathbf{n}\|}{\|\mathbf{n}\|} = r(h). \tag{5.7}$$

This formulation enables a more accurate representation of real-world poles, capturing both inclination and diameter variations along the pole's height.

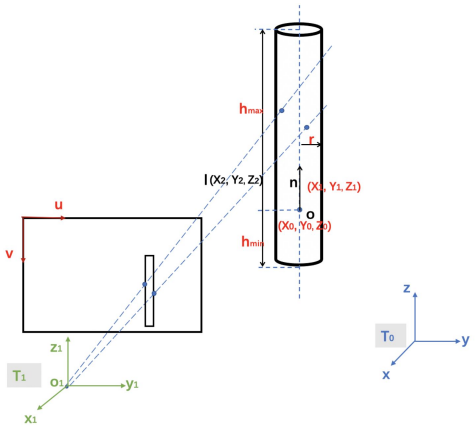
Parameter Optimization

Regardless of the chosen pole parametric model, optimization of model parameters follows a structured pipeline consisting of three main steps: initialization, refinement, and robust estimation of the top and bottom points.

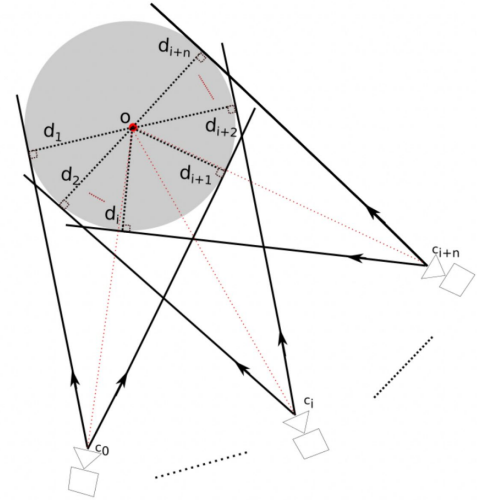
The initialization phase determines the center line axis $\mathbf{n} = (n_{\mathrm{x}}, n_{\mathrm{y}}, n_{\mathrm{z}})^{\mathrm{T}}$ using Principal Component Analysis (PCA). Since classical PCA relies on the empirical covariance matrix, it is highly sensitive to outliers. To enhance robustness against outliers, this thesis adopts the Minimum Covariance Determinant (MCD) estimator, which provides robust covariance estimation particularly suitable for lower-dimensional 3D point cloud data. The FAST-MCD algorithm [Rou99] is employed for computational efficiency. The robust PCA result is obtained from the eigenvectors of the estimated robust covariance matrix. Subsequently, the center point $\mathbf{o} = (x_{\mathrm{o}}, y_{\mathrm{o}}, z_{\mathrm{o}})^{\mathrm{T}}$ and the radius r are initially optimized using the accumulated landmark point cloud. This preliminary optimization minimizes the distance between a landmark point $\mathbf{p}$ and the pole surface model, expressed by the residual function:

$$\mathbf{r}_{\mathbf{p}} = \frac{\|(\mathbf{p} - \mathbf{o}) \times \mathbf{n}\|}{\|\mathbf{n}\|} - r. \tag{5.8}$$

Following initialization, optimization is performed using the accumulated landmark point cloud and the contours of the associated semantic instance masks.



(a) 3D visualization of the pole model optimization.



(b) 2D visualization of the pole model optimization.

Figure 5.16: Optimization of the pole model parameters. 3D viewing rays are projected onto the pole model surface using the contours of the semantic mask.

The contours are utilized to generate 3D viewing rays that extend from the camera sensor to the pole surface, leveraging the camera model, as illustrated in Figure 5.16. For a given viewing ray $\rho = \{\rho_s, \rho_d\}$, which is defined by a support point ρ_s and a direction vector ρ_d, the residual function is formulated as:

$$\mathbf{r}_\rho = \frac{\left|(\rho_s - \mathbf{o}) \cdot \left(\rho_d \times \mathbf{n}\right)\right|}{\left\|\rho_d \times \mathbf{n}\right\|} - r. \tag{5.9}$$

By combining Equation (5.8) and Equation (5.9) within a nonlinear optimization framework, the pole model parameters are refined. For the conical pole model, the constant radius r in the residual functions is replaced by the height-dependent radius function $r(h)$ from Equation (5.6), allowing for the optimization of both the base diameter r_o and the tapering slope s.

To determine the top and bottom limits of the pole, a reference plane is defined such that it passes through the center line axis $\mathbf{n}$ and is perpendicular to the vector from the camera sensor to the center point $\mathbf{o}$. The highest and lowest pixels of the pole in the semantic masks are then projected onto this reference plane, from which the maximum and minimum distances h_{max} and h_{min} are estimated. Since the bottom of a pole is often occluded in images, its position is determined by computing the intersection between the estimated center line axis $\mathbf{n}$ and the ground surface model. This ensures a more reliable estimation of the pole's true geometry, even in occlusion scenarios.

5.4.2 Traffic Light

Parametric Modelling

The geometric structure of traffic light primitives closely resembles that of pole primitives, with the primary distinction being that traffic lights are typically aligned perpendicular to the ground surface. In this thesis, it is assumed that the traffic light's center axis remains strictly vertical, simplifying its representation as a vertical cylinder with a fixed diameter. Additionally, an extra parameter is

introduced in the traffic light model compared to the pole model: the normalized orientation vector $\mathbf{v}_f$, which defines the facing direction of the traffic light. Illustrations of traffic light models with four distinct orientations are shown in Figure 5.17. The parameters presented in Figure 5.17 will be discussed in detail in the following sections.

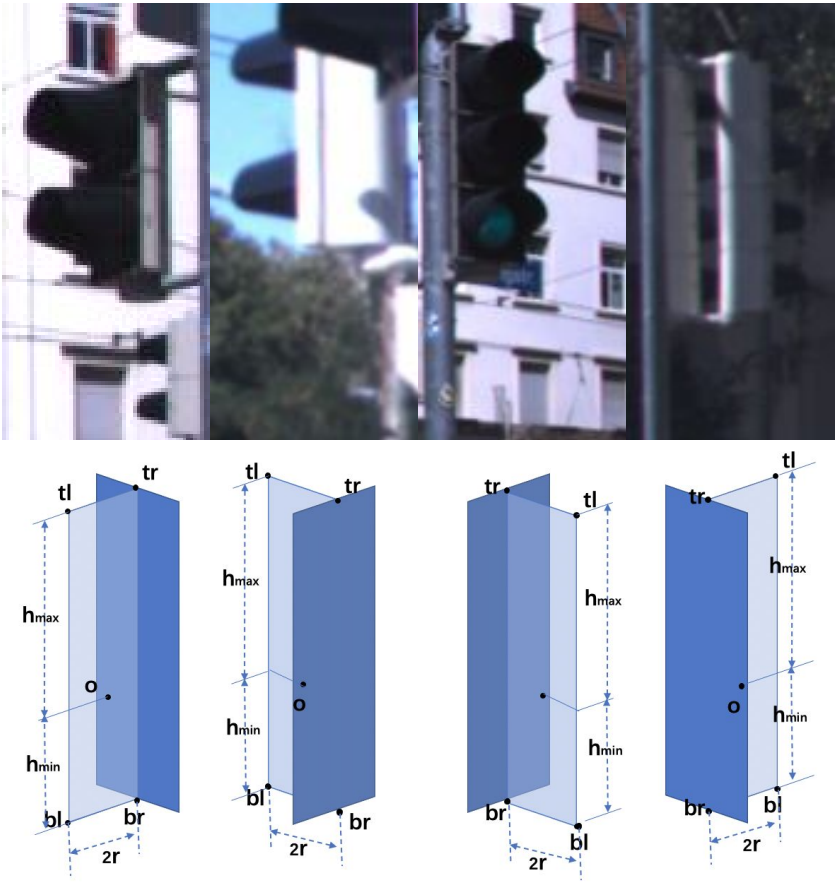


Figure 5.17: Parametric modeling of traffic lights in 3D space, categorized by different orientations in the camera sensor's front-view. The typical orientations, from left to right, include rear-left, front-left, rear-right, and front-right.

The parameters governing the traffic light's geometry, excluding the orientation vector $\mathbf{v}_f$, are referred to as geometric parameters. These parameters are identical to those of the pole model and are estimated using the approach described in Section 5.4.1. The estimation process relies solely on the accumulated landmark point cloud and the residual defined in Equation (5.8).

To determine the orientation vector $\mathbf{v}_f$, the estimated geometric parameters are leveraged to define two perpendicular rectangular planes in 3D space. As depicted in Figure 5.17, the dark blue rectangle corresponds to the display panel, while the light blue rectangle represents the visor panel. In this thesis, it is assumed that these two planar components are present in every traffic light instance. Without this assumption, estimating the traffic light's orientation using the approach introduced below would not be feasible.

The edge where the visor panel and the display panel intersect forms the rear boundary of the visor panel, with its upper and lower vertices denoted as **tr** and **br**, respectively. Conversely, the front edge of the visor panel has two corner vertices, labeled as **tl** and **bl**. Both the display and visor panels are assumed to have a width of $2 \cdot r$, where r represents the diameter of the cylindrical traffic light model. The center point **o** is determined from the cylinder model, and its center axis is constrained to remain perpendicular to the ground surface.

Parameter Optimization

The process of estimating the traffic light direction $\mathbf{v}_f$ is illustrated in Figure 5.18 and Figure 5.19. First, the four corner points of the visor rectangle must be located within the instance mask. As depicted in Figure 5.18, the corner points **tr** and **br** correspond to the vertices of the intersection between the projected visor panel and display panel rectangles in the instance mask. Conversely, the rectangle of the traffic light instance mask can be partitioned into two regions using corner points **tr** and **br**. These two partitions correspond to the visor panel and display panel projections in the semantic mask.

As illustrated in Figure 5.19, the bottom edge pixels of the bounding rectangle are traversed, and for each pixel, the ratio of pixels belonging to the instance

mask along a vertical scan line within the bounding rectangle is computed. Based on the computed ratio, the bounding rectangle is split into two subregions, **ABCD** and **BEFC**, and the center line **GH** of the rectangle **BEFC** is determined. The resulting segmentation corresponds to **AGHD** as the projected visor panel and **BEFC** as the projected display panel in the instance mask. The points **A**, **G**, **H**, and **D** correspond to the 3D points **tl**, **tr**, **br**, and **bl**, respectively. To refine the traffic light direction $\mathbf{v}_\mathrm{f}$, the optimization process minimizes the distances between the estimated 3D corners **tl**, **tr**, **br**, **bl** and their corresponding corners **A**, **G**, **H**, **D** within the instance mask images. The residual function considering a corner point **p** is formulated as follows:

$$
\begin{aligned}
\mathbf{n}_2 &= \mathbf{z}_\mathrm{unit} \times \mathbf{v}_\mathrm{f} \\
\lambda &= \frac{\mathbf{n}_2 \cdot (\mathbf{o} - \mathbf{o}_1)}{\mathbf{n}_2 \cdot \mathbf{n}_1} \\
\mathbf{p} &= \mathbf{o}_1 + \lambda \cdot \mathbf{n}_1 \\
\mathbf{r}_\mathrm{horizontal} &= \left|(\mathbf{p} - \mathbf{o}) \cdot \mathbf{v}_\mathrm{f}\right| - r \\
\mathbf{r}_\mathrm{vertical} &= \left|(\mathbf{p} - \mathbf{o}) \cdot \mathbf{n}\right| - h \\
\mathbf{r} &= (\mathbf{r}_\mathrm{horizontal}, \mathbf{r}_\mathrm{vertical})^\mathrm{T},
\end{aligned}
\tag{5.10}
$$

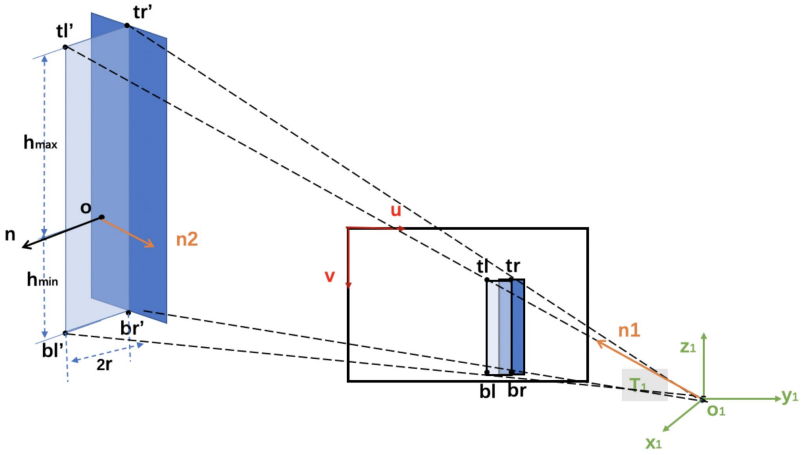


Figure 5.18: Optimization of the traffic light orientation.

where $\mathbf{o}_1$ represents the camera coordinate system origin, $\mathbf{n}_1$ is the normalized unit vector from $\mathbf{o}_1$ to the corner point $\mathbf{p}$, and $\mathbf{n}$ is the traffic light pole model center line unit vector. The variable $\mathbf{o}$ denotes the center of the visor rectangle, and $\mathbf{v}_\mathrm{f}$ is the traffic light unit direction vector. The term $\mathbf{n}_1$ represents also the 3D viewing ray direction from the camera sensor to the corner point $\mathbf{p}$, which is computed based on the camera model and the pixel coordinates of the corner point in the image. The radius of the traffic light cylinder is denoted as r, while h is the height difference from the corner point $\mathbf{p}$ to the visor rectangle center $\mathbf{o}$. Specifically, $h = h_\mathrm{max}$ for the upper corner points **tr** and **tl**, while $h = h_\mathrm{min}$ for the lower corner points **br** and **bl**. The residual vector $\mathbf{r}$ consists of two components: the horizontal residual $\mathbf{r}_\mathrm{horizontal}$ and the vertical residual $\mathbf{r}_\mathrm{vertical}$. The horizontal residual $\mathbf{r}_\mathrm{horizontal}$ quantifies the deviation of the corner point $\mathbf{p}$ from the expected horizontal distance r along the traffic light facing direction $\mathbf{v}_\mathrm{f}$. The vertical residual $\mathbf{r}_\mathrm{vertical}$ measures the deviation of the corner point $\mathbf{p}$ from the expected vertical distance h along the traffic light center line $\mathbf{n}$.

To parameterize the traffic light direction $\mathbf{v}_\mathrm{f}$, a naive approach would be to use a three-component 3D vector. However, given the assumption that traffic light center lines are perpendicular to the ground surface, the direction $\mathbf{v}_\mathrm{f}$ lies parallel to the xy-plane. Thus, a more compact representation is achieved by encoding the direction using a single angle parameter χ:

$$\mathbf{v}_\mathrm{f} = (\cos(\chi), \sin(\chi), 0)^\mathrm{T}. \tag{5.11}$$

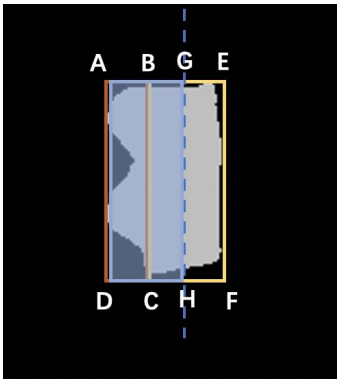


Figure 5.19: Subdivision of the bounding rectangle for optimization.

Since the angle χ is constrained within the range $[0, 2\pi)$ and lacks a robust initialization method, a multi-start optimization strategy is employed to ensure convergence to the global optimum. In this thesis, the angle χ is initialized with increments of $\pi/12$ across the interval $[0, 2\pi)$, yielding 24 independent optimization runs. The optimal traffic light orientation $\mathbf{v}_\mathrm{f}$ is selected as the solution yielding the lowest residual error after optimization.

5.4.3 Traffic Sign

As depicted in Figure 5.20, traffic signs exhibit a variety of geometric shapes, including circles, triangles, rectangles, and other irregular forms. To achieve a parametric representation of traffic signs across different shapes, distinct geometric models are formulated. In general, traffic signs are modeled on a 2D reference plane embedded in 3D space. This reference plane is characterized by a normal vector $\mathbf{n}_\mathrm{ref} = (n_\mathrm{ref,x}, n_\mathrm{ref,y}, n_\mathrm{ref,z})^\mathrm{T}$ and a reference center point $\mathbf{o}_\mathrm{ref} = (o_\mathrm{ref,x}, o_\mathrm{ref,y}, o_\mathrm{ref,z})^\mathrm{T}$. Each traffic sign is defined by a center point $\mathbf{o} = (x_\mathrm{o}, y_\mathrm{o})^\mathrm{T}$ on the reference plane. Furthermore, specific shape parameters are introduced depending on the geometric category of the traffic sign, such as the radius r for circular signs, the perpendicular distance of the side from the center d for triangular signs, and the half-width w and half-height h for rectangular and other non-standard signs. Additionally, traffic signs may undergo in-plane rotation about their center point $\mathbf{o}_\mathrm{ref}$, parameterized by a rotation angle χ.

The initial estimation of traffic sign model parameters is conducted using the accumulated landmark point cloud from the primitive association process. These parameter estimates are subsequently refined through an optimization framework that leverages both the spatial information from the landmark point cloud and the visual constraints from the associated instance semantic masks. The overall procedure for traffic sign parametric modeling can be summarized as follows: First, the parameters defining the reference plane, namely the normal

vector $\mathbf{n}_{\text{ref}}$ and the reference center point $\mathbf{o}_{\text{ref}}$, are estimated using the accumulated landmark point cloud via the robust PCA method introduced in Section 5.4.1, ensuring reliable initialization even with outliers. Next, the accumulated landmark LiDAR points are projected onto the reference plane, facilitating initialization of the traffic sign center point $\mathbf{o}$ and relevant shape parameters for each traffic sign category. Following this initialization, the contour pixels from the associated instance semantic masks are projected onto the reference plane using the camera model parameters. These projected contour points are employed within a nonlinear optimization framework to refine the traffic sign model parameters, ensuring accurate geometric representation and improved alignment with the visual data.

(a) Circle. **(b)** Triangle. **(c)** Rectangle. **(d)** Other shape.

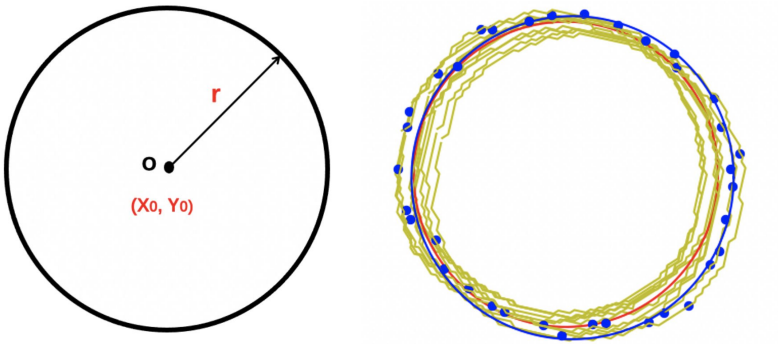
Figure 5.20: Examples of traffic signs with different geometric shapes.

Circular Traffic Signs

Figure 5.21a illustrates the parametric modeling of circular traffic signs. In addition to the reference plane parameters, two shape parameters are required to represent a circular traffic sign: the center point $\mathbf{o}$ on the reference plane and the radius r. Every 2D point $\mathbf{p} = (x, y)^{\mathrm{T}}$ lying on the circular perimeter of the traffic sign satisfies the following equation:

$$\|\mathbf{p} - \mathbf{o}\| = r. \tag{5.12}$$

As shown in Figure 5.21b, the initial shape parameters of the circular traffic sign are estimated by projecting the accumulated landmark point cloud onto the reference plane. The center point $\mathbf{o}$ is initialized as the centroid of the projected LiDAR points. Using this center point, the angular position of each projected point is computed. Based on these angular values, the projected points are partitioned into multiple angle-based groups. In this thesis, an angular interval of $\pi/18$ is used, resulting in 36 distinct groups. For each group, the point with the greatest distance from the center point $\mathbf{o}$ is selected as an edge point. These edge points, which are used for the radius estimation, are depicted as thick blue



(a) Parametric modeling. **(b)** Model parameters estimation.

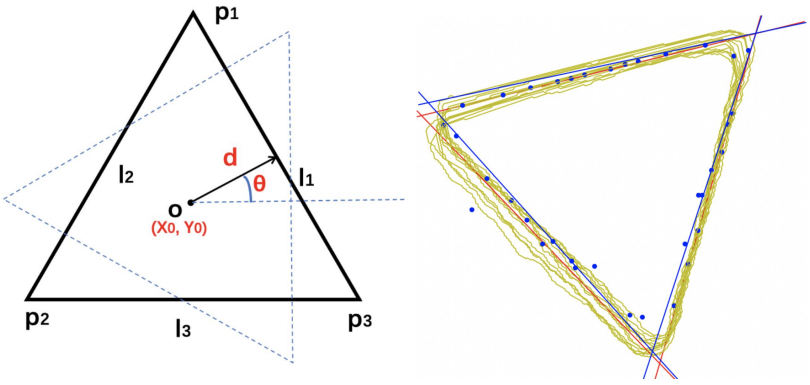
Figure 5.21: Parametric modeling and estimation for a circular traffic sign. In 5.21b: the blue points represent the LiDAR contour points projected onto the reference plane, while the yellow points correspond to the projected contour pixels from the instance semantic mask. The blue circle indicates the initialized model, and the red circle represents the optimized model projected onto the reference plane.

points in Figure 5.21b. The initial radius r is determined as the mean of the distances from the selected edge points to the center point $\mathbf{o}$. The resulting initialized model is visualized as the blue circle in Figure 5.21b.

Next, contour pixels of the associated instance semantic masks are projected onto the reference plane, as shown by the thin yellow points in Figure 5.21b. To refine the model parameters, an optimization process is applied, minimizing the deviation between the estimated radius r and the distances from the projected mask contour points to the center point $\mathbf{o}$. The residual function for a single projected mask contour point $\mathbf{p}$ is defined as:

$$\mathbf{r} = \|\mathbf{p} - \mathbf{o}\| - r. \tag{5.13}$$

Following the optimization, the refined model parameters yield the optimized circular representation, which is depicted as the red circle in Figure 5.21b.



(a) Parametric modeling. **(b)** Model parameters estimation.

Figure 5.22: Parametric modeling and estimation for a triangular traffic sign. In 5.22b: blue points represent the LiDAR contour points projected onto the reference plane, while yellow points denote the projected contour pixels from the tracked instance mask. The blue triangle represents the initialized model, whereas the red triangle corresponds to the optimized model projected onto the reference plane.

Triangular Traffic Signs

A triangular traffic sign is defined by three primary parameters: the center point $\mathbf{o}$, the distance d from the center point $\mathbf{o}$ to the triangle's side lines, and a rotation angle χ, which represents the triangle's orientation on the reference plane. Figure 5.22a illustrates the parametric modeling of triangular traffic signs. Points located on the three sides of the triangle satisfy the following relationships:

$$\begin{aligned} \text{line1:} \quad & d = \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi), \sin(\chi))^{\mathrm{T}}\| \\ \text{line2:} \quad & d = \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi + 3\pi/3), \sin(\chi + 3\pi/3))^{\mathrm{T}}\| \\ \text{line3:} \quad & d = \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi + 5\pi/3), \sin(\chi + 5\pi/3))^{\mathrm{T}}\|, \end{aligned} \tag{5.14}$$

which indicates that the distance d from the center point $\mathbf{o}$ to each triangle side can be determined by projecting the vector from the center point $\mathbf{o}$ to any point $\mathbf{p}$ on the side onto the direction perpendicular to that side.

To initialize the model parameters for a triangular traffic sign, a method similar to the initialization of circular traffic signs is applied. The accumulated landmark point cloud is used to estimate the triangle's edges. As depicted in Figure 5.22b, a Hough Transform-based line detection approach is utilized to identify the three side lines of the triangular traffic sign. Once the side lines are detected, their intersection points are computed. From these intersection points, the center point $\mathbf{o}$, the distance d, and the rotation angle χ are estimated.

To refine the estimated model parameters, the contour pixels of the associated instance semantic masks are projected onto the reference plane, as illustrated by the yellow points in Figure 5.22b. The optimization process involves adjusting the estimated triangle's shape parameters so that the projected mask contour points minimize their distances to the triangle's three side lines. For a single projected mask contour point $\mathbf{p}$, the residual function is formulated as:

$$\begin{aligned} \text{line1:} \quad & \mathbf{r}_1 = d - \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi), \sin(\chi))^{\mathrm{T}}\| \\ \text{line2:} \quad & \mathbf{r}_2 = d - \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi + 3\pi/3), \sin(\chi + 3\pi/3))^{\mathrm{T}}\| \\ \text{line3:} \quad & \mathbf{r}_3 = d - \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi + 5\pi/3), \sin(\chi + 5\pi/3))^{\mathrm{T}}\| \\ & \mathbf{r} = \min(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3). \end{aligned} \tag{5.15}$$

Following optimization, the refined triangular traffic sign model is shown as the red triangle in Figure 5.22b.

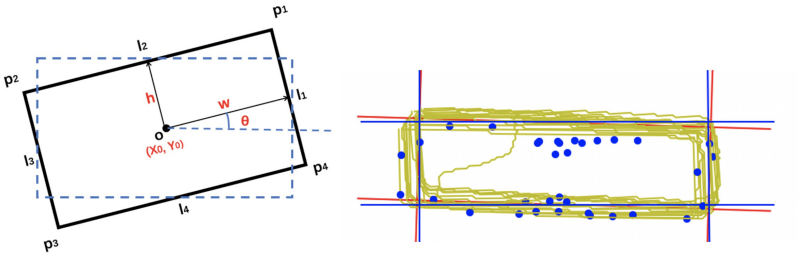
Rectangular Traffic Signs

A rectangular traffic sign is characterized by five parameters: the center point $\mathbf{o}$, the half width w, the half height h, and a rotation angle χ, which defines the sign's orientation on the reference plane. Figure 5.23a illustrates the parametric modeling of rectangular traffic signs. A point $\mathbf{p}$ located on any of the four rectangle sides satisfies the following relationship:

$$\begin{aligned}
&\text{line1:} \quad w = \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi), \sin(\chi))^{\mathrm{T}}\| \\
&\text{line2:} \quad h = \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi+\pi/2), \sin(\chi+\pi/2))^{\mathrm{T}}\| \\
&\text{line3:} \quad w = \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi+\pi), \sin(\chi+\pi))^{\mathrm{T}}\| \\
&\text{line4:} \quad h = \|(\mathbf{p}-\mathbf{o}) \cdot (\cos(\chi+3\pi/2), \sin(\chi+3\pi/2))^{\mathrm{T}}\|,
\end{aligned} \tag{5.16}$$

which indicates that the distances w and h from the center point $\mathbf{o}$ to the respective rectangle edges are obtained by projecting the vector from $\mathbf{o}$ to a point $\mathbf{p}$ on the side onto the direction perpendicular to that side.

To initialize the shape parameters of a rectangular traffic sign, a Hough Transform-based line detection method is employed to detect the four edges



(a) Parametric modeling. **(b)** Model parameter estimation.

Figure 5.23: Parametric modeling and estimation for a rectangular traffic sign. In 5.23b: blue points represent the LiDAR contour points projected onto the reference plane, while yellow points denote the projected contour pixels from the tracked instance mask. The blue rectangle corresponds to the initialized model, whereas the red rectangle represents the optimized model projected onto the reference plane.

of the rectangle. Once the four lines are identified, their intersections are computed to determine the rectangle's four corners. From these computed rectangle corner points, the center point $\mathbf{o}$, the half width w, the half height h, and the rotation angle χ can be easily estimated.

To refine the estimated parameters, the projected contour points from the associated instance semantic masks are utilized, as depicted by the yellow points in Figure 5.23b. The optimization process aims to minimize the distance between the projected mask contour points and the four edges of the rectangle. For a single projected mask contour point $\mathbf{p}$, the residual function is formulated as:

$$\begin{aligned}
\text{line1:} \quad & \mathbf{r}_1 = w - \|(\mathbf{p} - \mathbf{o}) \cdot (\cos(\chi), \sin(\chi))^{\mathrm{T}}\| \\
\text{line2:} \quad & \mathbf{r}_2 = h - \|(\mathbf{p} - \mathbf{o}) \cdot (\cos(\chi + \pi/2), \sin(\chi + \pi/2))^{\mathrm{T}}\| \\
\text{line3:} \quad & \mathbf{r}_3 = w - \|(\mathbf{p} - \mathbf{o}) \cdot (\cos(\chi + \pi), \sin(\chi + \pi))^{\mathrm{T}}\| \\
\text{line4:} \quad & \mathbf{r}_4 = h - \|(\mathbf{p} - \mathbf{o}) \cdot (\cos(\chi + 3\pi/2), \sin(\chi + 3\pi/2))^{\mathrm{T}}\| \\
& \mathbf{r} = \min(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4).
\end{aligned} \tag{5.17}$$

The optimized model is depicted as the red rectangle in Figure 5.23b.

Other Traffic Signs

Traffic signs that do not conform to the three previously discussed geometric shapes are categorized as other-shape traffic signs in this thesis. Due to their complex shapes, these traffic signs are approximated as rectangles for simplification. Thus, they are represented by the same parameters as rectangular traffic signs: the center point $\mathbf{o}$, the half-width w, the half-height h, and the rotation angle χ around the center point $\mathbf{o}$.

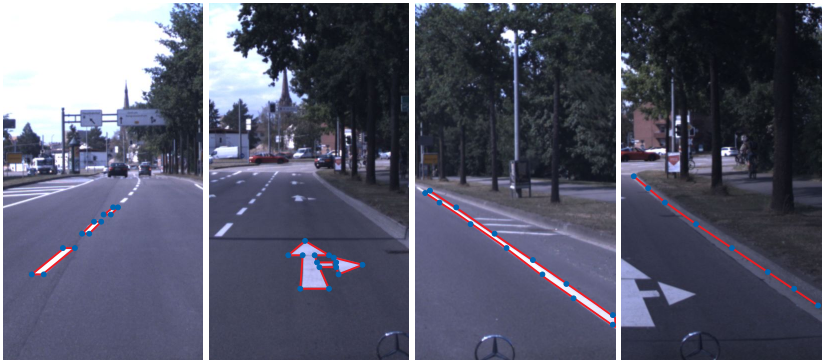
Initialization and optimization of the model parameters follow the same approach as that used for rectangular traffic signs. The only distinction lies in the irregular shape of the single instance mask corresponding to these traffic signs. To address this, a minimum rotated rectangle detection algorithm, implemented in *OpenCV* by Bradski [Bra00], is applied to approximate the complex traffic sign shape as a rectangle. The detected minimum rotated rectangle is then utilized as the semantic mask for subsequent optimization.

In principle, a free-form estimation approach analogous to the road marking method in Section 5.5 could be applied here as well, but is not pursued in this thesis since the rectangular approximation is sufficient for the intended map usage. Extending other-shape traffic sign modeling to a free-form representation is left as future work.

5.5 Parametric Mapping of Road Markings

5.5.1 Parametric Modelling

As illustrated in Figure 5.24, road marking primitives such as dashed lines, arrows, solid lines, and curbs exhibit distinct geometric characteristics. To develop a generalized parametric representation that accommodates these diverse shapes, a novel parametric modeling strategy is introduced in this thesis. This strategy must satisfy two essential requirements: first, it should enable the initialization and optimization of model parameters in a unified manner across different road marking types. Second, it should facilitate a precise geometric representation of road markings in a parametric form.



(a) Dashed line. **(b)** Arrow. **(c)** Solid line. **(d)** Curb.

Figure 5.24: Parametric modelling of road marking primitives with different geometric shapes. The blue points represent the possible model points and the red lines indicate the connectivity between the model points.

Dashed lines, arrows, and solid lines are modeled as closed 3D polygons, while curbs are represented as open 3D polylines. Exemplary parametric models are depicted in Figure 5.24. The point allocations adopted are: four points for dashed lines (corresponding to physical corner points), fifteen points for arrows, and thirty points for solid lines and curbs. These allocations balance model complexity and representation accuracy for efficient yet precise parametric description.

5.5.2 Vectorization

The vectorization process transforms the accumulated road marking landmark point cloud into ordered 3D landmark edge points that define the primitive's shape. Similar to elevated primitive initialization, vectorization serves as an initialization step where model points are determined using either a 3D polygon or 3D polyline. For curbs (3D polylines), model points are sampled from the landmark point cloud along the ego vehicle's driving direction. For dashed lines, arrows, and solid lines (3D polygons), vectorization consists of two key steps: concave hull detection and polygon simplification.

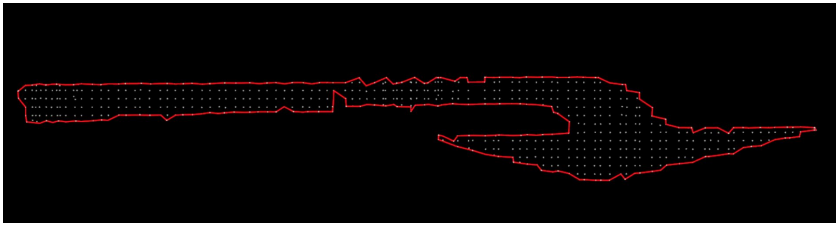
Concave Hull Detection

For dashed lines, arrows, and solid lines, the concave hull is detected using the K-Nearest Neighbors (KNN)-based approach introduced by Moreira et al. [Mor07]. This algorithm constructs the concave hull by iteratively selecting points from the accumulated landmark point cloud while ensuring geometric consistency. The concave hull detection process follows five key steps:

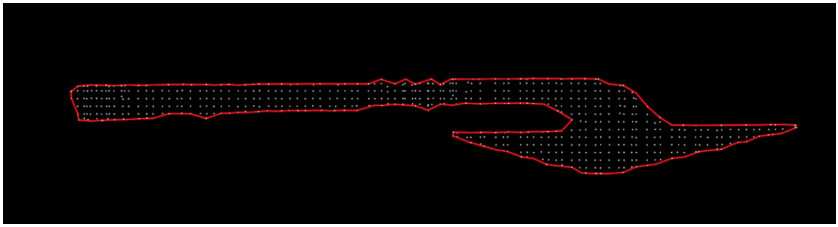
- **Initialization**: The process begins with the selection of an initial point, serving as the starting reference for constructing the hull.
- **KNN Search**: The algorithm identifies the KNN of the current point to determine candidate points for hull expansion.

- **Next Point Selection**: Among the KNN, the algorithm selects the point that forms the largest right-hand angle with the preceding edge while ensuring no intersection with existing edges.
- **Edge Validation and Adjustment**: To prevent self-intersections, each newly formed edge is checked against previously generated edges. If an intersection is detected, the current k value is deemed insufficient, requiring an increase in k before restarting the process.
- **Completion Check**: The process iterates until the hull is closed, ensuring all points belong to the hull or remain inside it.

Two exemplary results of concave hull detection are illustrated in Figure 5.25 for an arrow. A suboptimal choice of k results in a jagged contour, as shown in Figure 5.25a, while increasing k smooths the contour, as shown in Figure 5.25b. Progressively increasing k leads to a gradual reduction in the number of edge points until it stabilizes at a constant value. To determine whether the number of edge points has converged, a smoothness criterion is introduced based on the



(a) k = 5.



(b) k = 8.

Figure 5.25: Exemplary results of concave hull detection for an arrow with different hyperparameter settings. The gray points represent the accumulated landmark point cloud, while the red lines depict the connections between the detected edge points.

relative change in the number of edge points:

$$\frac{N(n+1) - N(n)}{N(n)} < q_{\text{smooth}}, \tag{5.18}$$

where $N(n)$ and $N(n+1)$ denote the number of edge points obtained using k values of n and $n+1$, respectively, and q_{smooth} is the smoothness threshold (set to 0.08). Using this criterion, the optimal k value is determined iteratively until convergence is achieved.

Polygon Simplification

The concave hull detection process results in an ordered set of 3D edge points defining road marking primitives. However, the number of edge points is inconsistent and depends on the hyper-parameter k, which cannot be directly controlled. To ensure a fixed number of edge points, as defined in Section 5.5.1, a polygon simplification step is required. The objective of polygon simplification is to reduce the number of edge points while preserving the geometric structure of the original polygon as accurately as possible.

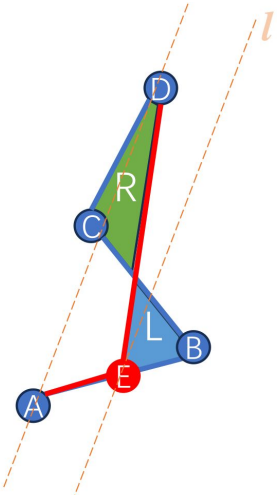
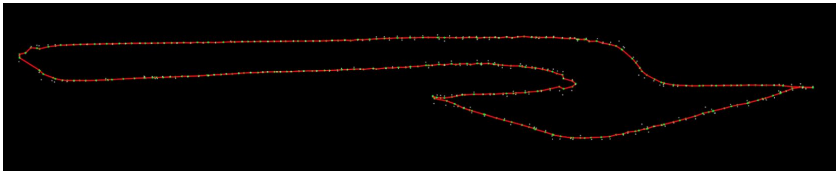


Figure 5.26: The segment collapse algorithm for polygon simplification.

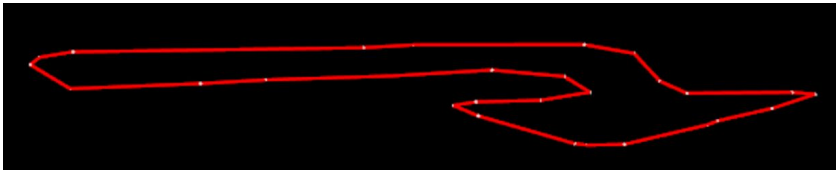
For polygon simplification, this thesis applies the area-preserving segment collapse algorithm introduced by Kronenfeld et al. [Kro20]. Any polygon with n vertices can be represented by n segments consisting of four adjacent points, where each segment can be approximated using three ordered points. As depicted in Figure 5.26, a segment defined by an ordered point sequence **ABCD** can be simplified to a sequence **AED**. This transformation results in two enclosed regions: region $\mathscr{A}_{\mathrm{R}}$ (green) and region $\mathscr{A}_{\mathrm{L}}$ (blue). The combined area of these regions is referred to as the areal displacement.

Since the area-preserving segment collapse algorithm is sensitive to high-frequency noise in polygon points, a smoothing step is necessary before polygon simplification. In this thesis, the bilateral filter introduced by Paris et al. [Par07] is applied to smooth the polygon before simplification. For each point $\mathbf{p}$ in the polygon, the local normal vector $\mathbf{n_p}$ is computed using its neighboring points. The position of $\mathbf{p}$ is then updated by smoothing the Euclidean distance and the projection distance along $\mathbf{n_p}$ relative to its neighboring points.

The original bilateral filter was designed for unstructured points and uses the Euclidean distance for neighborhood searching. However, since polygon points are ordered and have known topological relationships, incorporating these relationships into the bilateral filter improves smoothing performance.



(a) Smoothed polygon.



(b) Simplified polygon.

Figure 5.27: Exemplary results of polygon simplification for an arrow. 5.27a shows the smoothed polygon using bilateral filtering. 5.27b shows the final simplified polygon.

During the simplification process, each collapse operation is performed on the segment with the smallest areal displacement among all available segments. After each collapse, the affected segments are updated, and the process continues until the predefined number of edge points is reached. The optimization follows two key principles, illustrated in Figure 5.26:

- **Area-preserving principle**: The areas of regions $\mathscr{A}_{\mathrm{R}}$ and $\mathscr{A}_{\mathrm{L}}$ should be equal to prevent shape distortions. The new point **E** must lie on a line l parallel to segment **AD**.
- **Minimizing areal displacement principle**: To minimize areal displacement, following Kronenfeld et al. [Kro20], the optimal position of **E** is at the intersection of line l with either line **AB** or line **CD**.

Applying polygon simplification to the smoothed polygon yields a consistent and geometrically accurate polygon representation. An exemplary result of the polygon simplification process for an arrow is shown in Figure 5.27.

5.5.3 Optimization

Figure 5.28 presents four examples of vectorized road-marking primitives with initially estimated model points back-projected into camera images. These projections show that the initial model points remain suboptimal and do not fully align with the corresponding road-marking shapes. To enhance parameter estimation accuracy, an optimization step refines these vectorized models using the associated instance semantic masks.

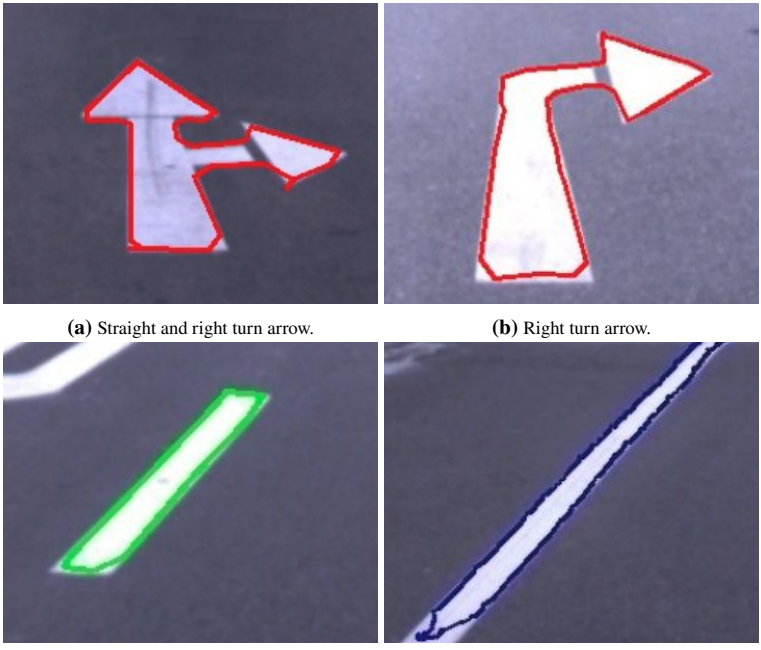
The residual function for road-marking optimization follows the approach used for traffic sign optimization in Section 5.4.3. Instance masks are projected onto the ground surface, and the distance between projected points and modeled shape line connections is minimized, similar to the optimization processes detailed in Equation (5.15) and Equation (5.17).

An alternative method for constructing the residual function involves utilizing a distance transform of the semantic segmentation image. The distance transform computes a map in which pixel values represent the Euclidean distance

to the nearest edge of the semantic mask. A distance transform example for a semantic mask is depicted in Figure 5.29. The 3D model points are projected into this distance transform map, and the corresponding pixel values are used to formulate the residual function:

$$\mathbf{r} = \mathscr{M}_{\text{dt}}(\pi(\mathbf{K}^{-1}\mathbf{T}_{\text{map}}^{\text{cam}}\mathbf{p})), \tag{5.19}$$

where $\mathscr{M}_{\text{dt}}$ denotes the distance transform map lookup table, π represents the camera projection function, $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{T}_{\text{map}}^{\text{cam}}$ is the transformation matrix from the map frame to the camera frame, and $\mathbf{p}$ is a 3D model point. By minimizing the distance transform value, the optimized 3D model points better align with the instance semantic mask, ensuring a more accurate geometric representation of the road marking primitive.



(a) Straight and right turn arrow. **(b)** Right turn arrow.

(c) Dashed line. **(d)** Solid line.

Figure 5.28: Examples of back-projected vectorized model points for road marking primitives.

While the model representation of road marking primitives consists of a set of relatively sparse model points in Figure 5.24, optimizing only these sparse points is suboptimal. Instead, the focus should be on optimizing the connections between model points, as they define the geometric shape of the primitive. To achieve this, additional intermediate points are sampled along the connections between model points, and these sampled points are projected onto the distance transform map. By incorporating these sampled points into the residual function, the entire shape of the road marking is optimized.

Another key aspect of the optimization process is ensuring that the cost function remains differentiable when using the distance transform lookup table $\mathscr{M}_{\mathrm{dt}}$. To achieve this, bilinear interpolation is applied within the local region of each projected point, ensuring a differentiable optimization process.

As depicted in Figure 5.24, road marking primitives exhibit certain geometric constraints, which can be used as regularization terms in the optimization. For instance, dashed lines, which are modeled as 3D polygons, should have intersection angles of approximately 90 degrees between adjacent segments. Similarly, solid lines and curbs should maintain an intersection angle of approximately 180 degrees. For a single angular regularization constraint η_{r}, the residual function can be defined as:

$$\mathbf{r} = \eta_{\mathrm{r}} - \eta, \tag{5.20}$$

where η_{r} represents the expected regularization angle, and η is the computed angle between adjacent line segments. By incorporating these geometric constraints, the optimized model not only fits the semantic mask but also maintains the expected structural integrity of the road marking primitive.

(a) Semantic mask. **(b)** Contour. **(c)** Distance transform map.

Figure 5.29: Example of distance transform mapping from a semantic mask of an arrow.

5.6 Mapping Example Results

The final outcome of the semantic parametric primitive mapping process for both elevated and road marking primitives consists of optimized parametric models represented in 3D space. Collectively, these parametric models form a structured 3D semantic parametric map.

In Figure 5.30, the generated semantic parametric maps for three representative driving scenarios are visualized. The elevated and road marking primitives are accurately and robustly modeled using the proposed parametric representation, ensuring consistency and precision in the mapped environment.

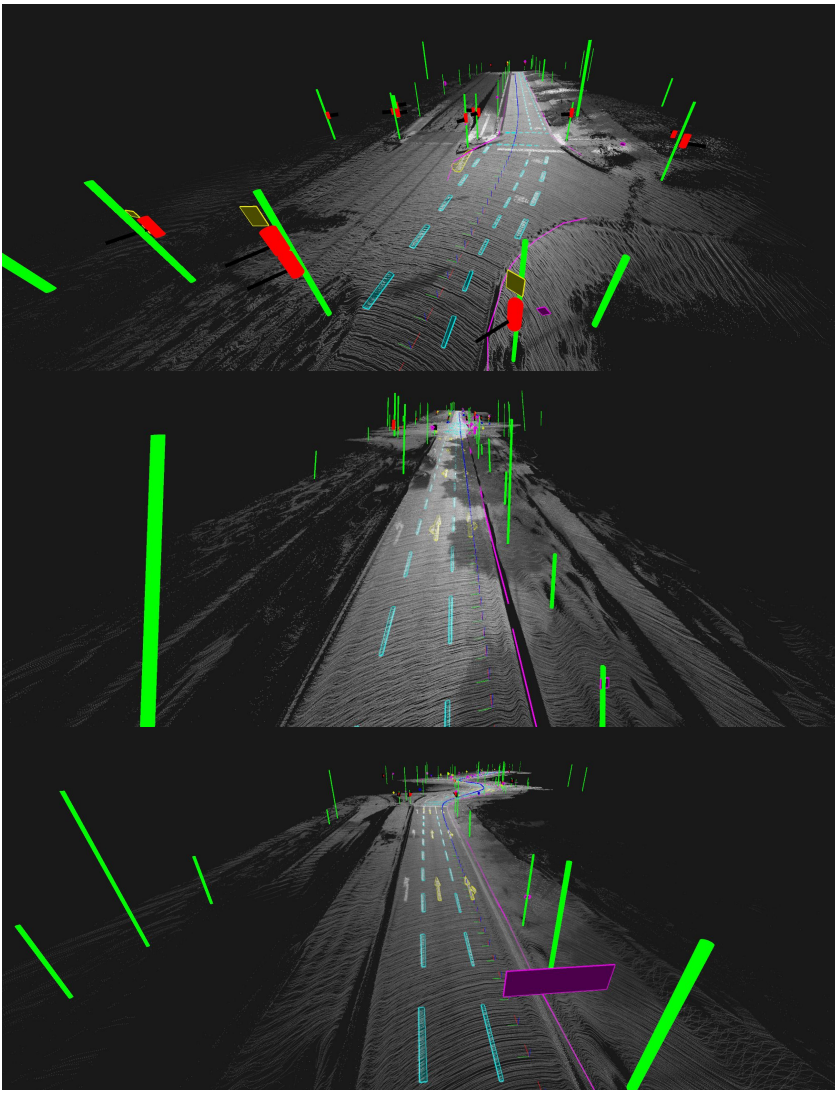


Figure 5.30: Exemplary results of the parametric mapping of elevated and road marking primitives for three typical driving scenarios. The parametric models are highlighted in distinct colors. The coordinate system frames on the ground surface indicate the ego-motion trajectory of the vehicle during data acquisition.

6 Evaluation

In this chapter, the proposed multi-modal continuous-time trajectory SLAM and semantic primitive parametric mapping framework are evaluated using a real-world dataset, which was recorded by an experimental vehicle in urban environments. The evaluation begins with a comprehensive presentation of the experimental vehicle's sensor setup, including the configuration of cameras, LiDAR, IMU, and GNSS. Details regarding sensor synchronization, triggering mechanisms, calibration procedures, and the characteristics of the evaluation dataset are provided. Two evaluation methods and their corresponding metrics are introduced: the accumulated LiDAR point cloud structure score for assessing the continuous-time trajectory SLAM performance, and the primitive mask Intersection over Union (IoU) for quantifying the accuracy of the semantic primitive parametric mapping. Both qualitative and quantitative evaluation results are presented and discussed in detail, demonstrating the effectiveness and accuracy of the proposed approaches.

6.1 Sensor Setup and Dataset

The evaluation dataset was collected using the experimental vehicle *Bertha One* Ziegler et al. [Zie14], as shown in Figure 6.1. This vehicle is equipped with the following sensors: a 360° rotated Velodyne Alpha Prime 128-layer LiDAR mounted on the roof, three Blackfly S gray cameras (two positioned at the front and one at the rear), a Blackfly S color camera at the front, a Xsens MTi-300 IMU located beneath the LiDAR, a Ublox NEO-M8T GNSS with an attached antenna on the roof, and integrated vehicle wheel odometry.

All sensors within the multi-modal setup are synchronized to the unified GNSS time reference, ensuring consistent temporal coordination throughout the data acquisition process. Synchronization and triggering signals are generated by a dedicated trigger box hardware system and distributed to all sensors. The acquisition rates of the sensor suite are configured through the trigger box: the LiDAR system operates at 10 Hz, the gray and color cameras operate at 10 Hz, the IMU operates at 100 Hz, the GNSS receiver operates at 5 Hz, and the wheel odometry system operates at 100 Hz.

Cameras are calibrated both intrinsically and extrinsically using the approach presented by Strauss [Str15]. Extrinsic calibration between the camera and LiDAR is performed using the method introduced by Kühner et al. [Küh19], Kümmerle [Küm20b], and Kümmerle et al. [Küm20a]. Minimal online refinement of the extrinsic parameters between the camera and LiDAR is conducted using the approach presented by Hu et al. [Hu22]. The rear axle is extrinsically calibrated relative to the sensor setup using the method presented by Kümmerle et al. [Küm19a]. The vehicle coordinate system is defined as the projection of the rear axle coordinate system onto the calibrated ground plane. The IMU is mounted directly beneath the LiDAR, and the extrinsic parameters between

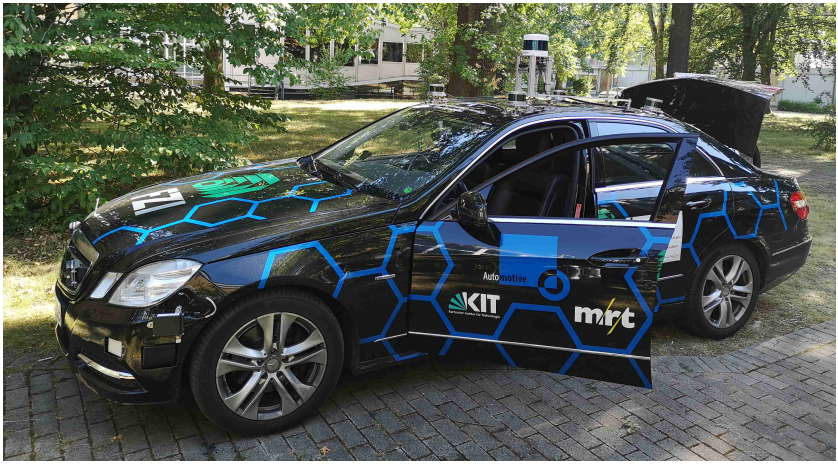
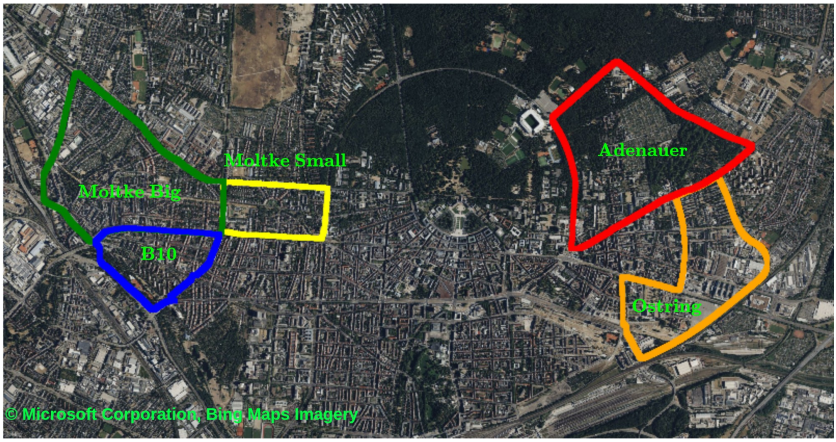
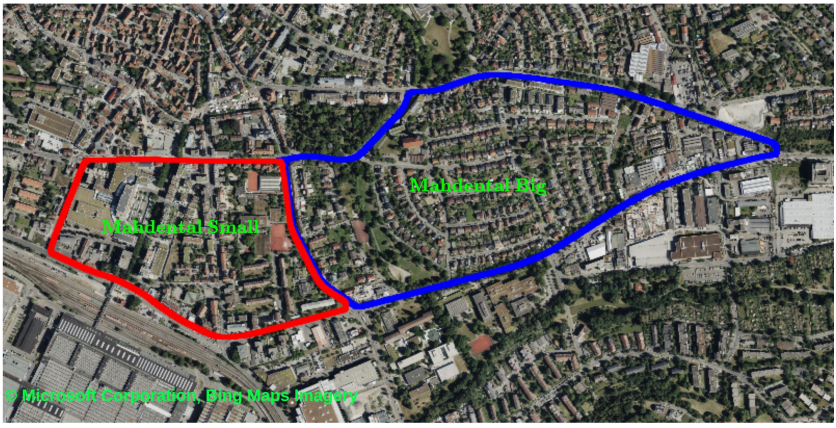


Figure 6.1: Sensor setup of the experimental vehicle *Bertha One* Ziegler et al. [Zie14].

the two are slightly adjusted using the motion-based Hand-Eye Calibration approach proposed by Horaud et al. [Hor95]. The GNSS is assumed to have a certain height offset relative to the vehicle coordinate system.



(a) 5 Sequences in Karlsruhe.



(b) 2 Sequences in Sindelfingen.

Figure 6.2: The evaluation dataset collected in Germany. Background imagery © Microsoft Corporation and its data providers (Bing Maps).

For data preprocessing, LiDAR point clouds are motion-compensated using motion estimates from the continuous-time trajectory SLAM approach discussed in Chapter 4. The continuous-time representation enables accurate pose estimation at arbitrary timestamps, facilitating precise motion compensation for each individual LiDAR point, since each LiDAR point has its own timestamp during a full rotation of the LiDAR sensor. For elevated primitives such as poles, traffic lights, and traffic signs, the seamless segmentation network introduced by Porzi et al. [Por19] generates instance masks. For road marking primitives including dashed lines, solid lines, arrows, and curbs, the semantic segmentation network introduced by Bieder et al. [Bie23] is employed. In the work presented by Bieder et al. [Bie23], the semantic segmentation networks are trained using automatically generated real world data by reprojecting HD map elements into the camera images using accurate vehicle poses.

The evaluation dataset was collected in the cities of Karlsruhe and Sindelfingen in Germany, as illustrated in Figure 6.2 using the experimental vehicle *Bertha One*. The recorded dataset mainly comprises urban driving scenarios, including various road types such as highways (part of sequence B10) and urban streets. These sequences are labeled as: Ostring (Ost), Adenauer (Ade), Moltke Small (Mos), Moltke Big (Mob), B10 (B10), Mahdental Small (Mas), and Mahdental Big (Mab). More detailed information is provided in Table 6.1.

Table 6.1: Evaluation dataset sequence information details including the pivot frame number, driving distance, driving hour, and sensor availability for each sequence. The dataset consists of seven sequences in total, approximately 28.63 km driving distance, 1.2 effective driving hours, about 43000 image and LiDAR sensor data frames. The wheel odometry, camera, LiDAR, GNSS, and IMU sensors are used in the dataset. For the IMU sensor, the data is not available for the sequences Mahdental Small and Mahdental Big.

Sequence	**Ade**	**B10**	**Mob**	**Mos**	**Ost**	**Mab**	**Mas**
Pivot Frame Number	5532	6661	10463	6462	7001	3689	3132
Driving Distance (km)	5.81	3.63	5.35	2.97	5.72	2.95	2.20
Driving Hour (seconds)	553	666	1046	646	700	369	313
Wheel Odometry	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓
LiDAR	✓	✓	✓	✓	✓	✓	✓
GNSS	✓	✓	✓	✓	✓	✓	✓
IMU	✓	✓	✓	✓	✓	-	-

6.2 Evaluation Methods

6.2.1 Qualitative Evaluation

To qualitatively assess the estimated motion trajectory, the accumulated point clouds are visually inspected. For the qualitative evaluation of semantic parametric mapping, the 3D maps are visualized and the mapped 3D primitives are reprojected onto the corresponding camera images, allowing for a visual comparison between the reprojected primitives and the actual camera images.

6.2.2 Quantitative Evaluation

Point Cloud Structure Score

In real-world scenarios, the ground truth trajectory is typically unavailable. To quantitatively evaluate the accuracy of motion estimation, a novel self-supervised evaluation metric is introduced: the structure score, which enables a relative assessment of the accumulated point cloud quality. For this evaluation, it is assumed that the sensors are well calibrated and that the calibration error is negligible compared to the trajectory estimation error. As outlined in Section 4.2.4, point clouds are categorized into three classes based on the geometric properties of the points within a voxel grid: edge points, surface

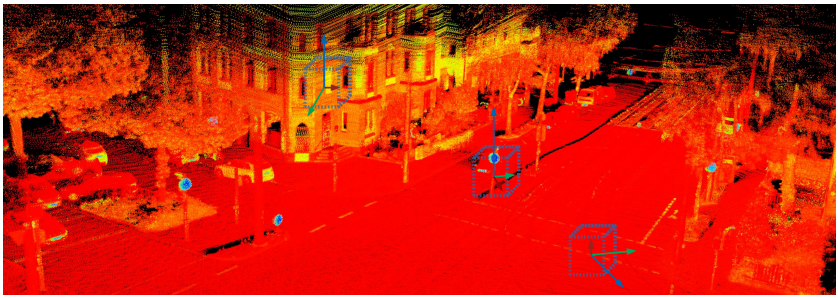


Figure 6.3: Covariance matrix eigenvalues for three representative grid voxels: edge, surface, and ground. The structure score is computed as the ratio of the eigenvalues, sorted in ascending order: λ_1 (red) $\leq \lambda_2$ (green) $\leq \lambda_3$ (blue).

points, and ground points. The structure score is computed using the eigenvalue ratio of the covariance matrix of the accumulated point cloud within a voxel grid, as illustrated in Figure 6.3. For edge feature voxels, a linearity score S_{L} is employed to quantify the degree to which the point cloud structure within the voxel exhibits line-like characteristics. For surface and ground feature voxels, a planarity score S_{P} is utilized to assess the extent to which the structure demonstrates planar properties. To obtain a unified scalar metric for comparative analysis, the structure score integrates the linearity score S_{L} and the planarity score S_{P} as follows:

$$S_{\mathrm{s}} = \begin{cases} S_{\mathrm{L}} = \frac{\lambda_3 - \lambda_2}{\lambda_3}, & \text{if edge voxel} \\ S_{\mathrm{P}} = \frac{\lambda_2 - \lambda_1}{\lambda_3}, & \text{if surface/ ground voxel,} \end{cases} \tag{6.1}$$

where λ_1, λ_2, and λ_3 are the eigenvalues of the covariance matrix, sorted in ascending order. The structure score serves as a reliable indicator of the accumulated point cloud quality and the accuracy of trajectory estimation. It ranges from 0 to 1, where a higher score indicates a more structured and accurate point cloud, which in turn reflects a more accurate trajectory estimation.

Primitive Mask mean IoU Score

Since labeling 3D primitives is inherently challenging, and human labeling accuracy may be lower than the mapping accuracy, we employ the IoU score as a quantitative evaluation metric for semantic parametric mapping. The IoU score quantifies the overlap between the reprojected primitives and their associated camera semantic masks, which is defined as follows:

$$S_{\mathrm{iou}} = \frac{\mathscr{A}_{\mathrm{o}}}{\mathscr{A}_{\mathrm{u}}}, \tag{6.2}$$

where $\mathscr{A}_{\mathrm{o}}$ represents the overlapping area, and $\mathscr{A}_{\mathrm{u}}$ denotes the union area. The IoU score ranges from 0 to 1, where a higher score indicates better alignment between the reprojected primitives and the camera semantic masks. Under the mean IoU, the IoU score is averaged over all reprojected landmark primitives, which can be further averaged across frames, classes, and sequences.

6.3 Continuous-Time SLAM

In this section, the continuous-time trajectory SLAM is evaluated. In prior work presented by Sons [Son21], visual graph SLAM was applied exclusively to camera data. In this thesis, it is extended by incorporating both camera and LiDAR data. By comparing the results of visual and visual-LiDAR graph SLAM, the impact of incorporating LiDAR data on trajectory estimation is assessed. By comparing graph-based and continuous-time SLAM results using camera, LiDAR, and IMU data, the influence of the continuous-time trajectory representation on trajectory estimation is evaluated. For the sequences Mahdental Big and Mahdental Small, the evaluation is conducted only using visual SLAM and visual-LiDAR graph SLAM, because the IMU data is not available. For other sequences, the evaluation is conducted using the visual graph SLAM, visual-LiDAR graph SLAM, and continuous-time SLAM.

Accumulated Point Cloud

Exemplary accumulated point clouds obtained using the estimated trajectories from visual graph SLAM without and with scale optimization, visual-LiDAR graph SLAM, and continuous-time SLAM of the same driving scenario are shown in Figure 6.4. It is apparent that the accumulated point cloud generated using visual-LiDAR graph SLAM exhibits a more structured and detailed representation compared to that obtained using visual graph SLAM. The scale optimization of the visual graph SLAM also improves the structure and detail of the accumulated point cloud. Building facades, poles, traffic lights, traffic signs, road markings, and other infrastructure elements are more clearly distinguishable. This qualitative assessment suggests that the trajectory estimation is more accurate and robust when LiDAR point cloud data is incorporated into the pose graph optimization process. The additional geometric information provided by LiDAR contributes to enhanced structural fidelity and detail. Furthermore, it is evident that the accumulated point clouds generated using the trajectories from continuous-time SLAM exhibit structural similarity and comparable levels of detail to those obtained from visual-LiDAR graph SLAM.

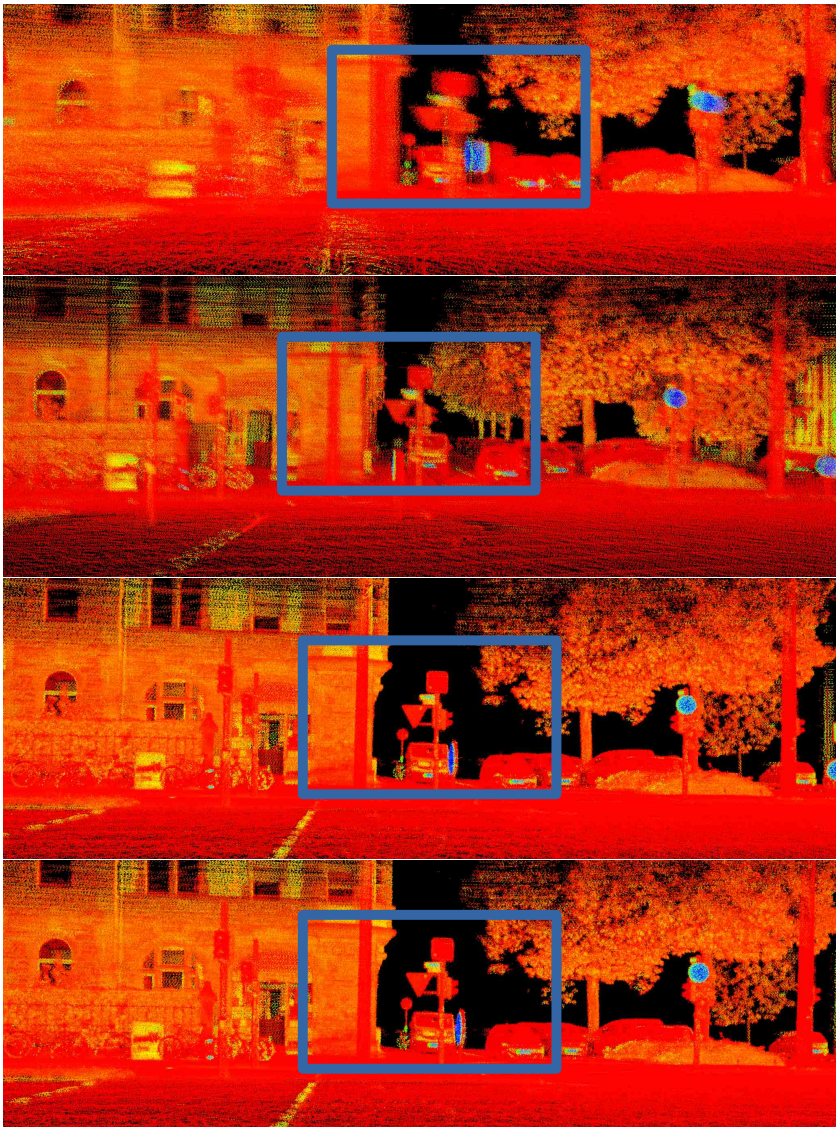


Figure 6.4: Exemplary accumulated point clouds using the estimated trajectory from visual graph SLAM, visual graph SLAM with scale optimization, visual-LiDAR graph SLAM and continuous-time SLAM (top to bottom). The blue boxes highlight regions, where differences between approaches are most evident.

Structure Score

To evaluate the SLAM approaches quantitatively, the structure scores of the accumulated LiDAR point clouds are computed using the trajectories estimated from visual graph SLAM without scale optimization, visual graph SLAM with scale optimization, visual-LiDAR graph SLAM, and continuous-time SLAM for all seven evaluation sequences. For the structure score computation, a voxel grid size of 0.5 $m \times 0.5\ m \times 0.5\ m$ is used, and LiDAR point clouds are accumulated in batches of 40 frames, corresponding to approximately 4 seconds of driving, and points beyond 50 m from the LiDAR sensor are excluded. The LiDAR point clouds used are motion-compensated, and for the accumulation, each point cloud is considered as a group with the same timestamp and the same pose. The results are aggregated across accumulation batches for each sequence and presented in Figure 6.5 as the box plot of the structure scores.

From the box plot in Figure 6.5, it is observed that the visual graph SLAM without scale optimization (orange) exhibits the lowest structure scores, while the visual graph SLAM with scale optimization (orange) shows significant improvement. The reason could be a scale drift in the visual graph SLAM resulting from a calibration error of the stereo camera setup. By applying GNSS measurements, a global scale parameter is estimated and the scale drift is corrected. The structure scores corresponding to visual-LiDAR graph SLAM are generally much higher compared to visual graph SLAM. This is primarily due to the integration of LiDAR point cloud data within the pose graph optimization process, which corrects the scale drift. To see the difference more clearly with numbers, the summarized statistics of the structure scores are presented in

Table 6.2: Summary statistics of the structure scores across all seven evaluation sequences using Quantile Q1(25%), Q2(50%), Q3(75%), and the mean value.
CO = Camera Only, COSO = Camera Only Scale Optimized, CL = Camera Lidar, CT = continuous-time.

	Q1(25%)	**Q2(50%)**	**Q3(75%)**	∅
CO	0.478	0.624	0.765	0.614
COSO	0.533(+11.5%)	0.661(+5.9%)	0.784(+2.5%)	0.654(+6.5%)
CL	0.549(+14.9%)	0.685(+9.8%)	0.811(+6.0%)	0.675(+9.9%)
CT	**0.551**(+15.3%)	**0.686**(+9.9%)	**0.812**(+6.1%)	**0.677**(+10.3%)

Table 6.2 as overall scores, which also shows that the structure scores of visual-LiDAR graph SLAM are much higher than those of visual graph SLAM. The structure scores of continuous-time SLAM are similar to those of visual-LiDAR graph SLAM, which is also evident in Figure 6.5.

However, do these similar structure scores necessarily indicate equivalent trajectory estimation accuracy between both approaches? During the structure score computation for Figure 6.5, only the motion at the LiDAR point cloud acquisition time is used, the estimation accuracy of the motion between those timestamps is not evaluated properly. To overcome this problem and evaluate the motion between timestamps, a new structure score computation is proposed, using the raw LiDAR point clouds without motion compensation. During the accumulation of LiDAR point clouds, the timestamp of each single LiDAR point is used to determine the corresponding pose, and the points are considered separately. With the continuous-time trajectory representation, the motion at any timestamp can be easily estimated and used directly, and with the pose graph representation, the motion between poses needs to be interpolated. This new structure score computation is done on the Ostring sequence without applying loop closure and the results are shown in Table 6.3. From the detailed comparison of the structure scores, it is observed that the structure scores of continuous-time SLAM are slightly higher than those of visual-LiDAR graph SLAM across all percentiles and mean values. This is primarily due to the continuous-time trajectory representation and the incorporation of IMU data. All of these structure score results are also consistent with the qualitative assessment of the accumulated point clouds presented in Section 6.3.

Table 6.3: The structure scores of the accumulated LiDAR point clouds using the estimated trajectories from visual-LiDAR graph SLAM, and continuous-time SLAM on the Ostring evaluation sequence by applying the raw not motion compensated LiDAR point clouds and consider every single LiDAR point separately without loop closure. The Quantile Q1(25%), Q2(50%), Q3(75%), and the mean value are reported.
CL = Camera Lidar, CT = continuous-time.

	Q1(25%)	**Q2(50%)**	**Q3(75%)**	∅
CL	0.560	0.694	0.815	0.683
CT	**0.575**(+2.7%)	**0.710**(+2.3%)	**0.829**(+1.7%)	**0.698**(+2.2%)

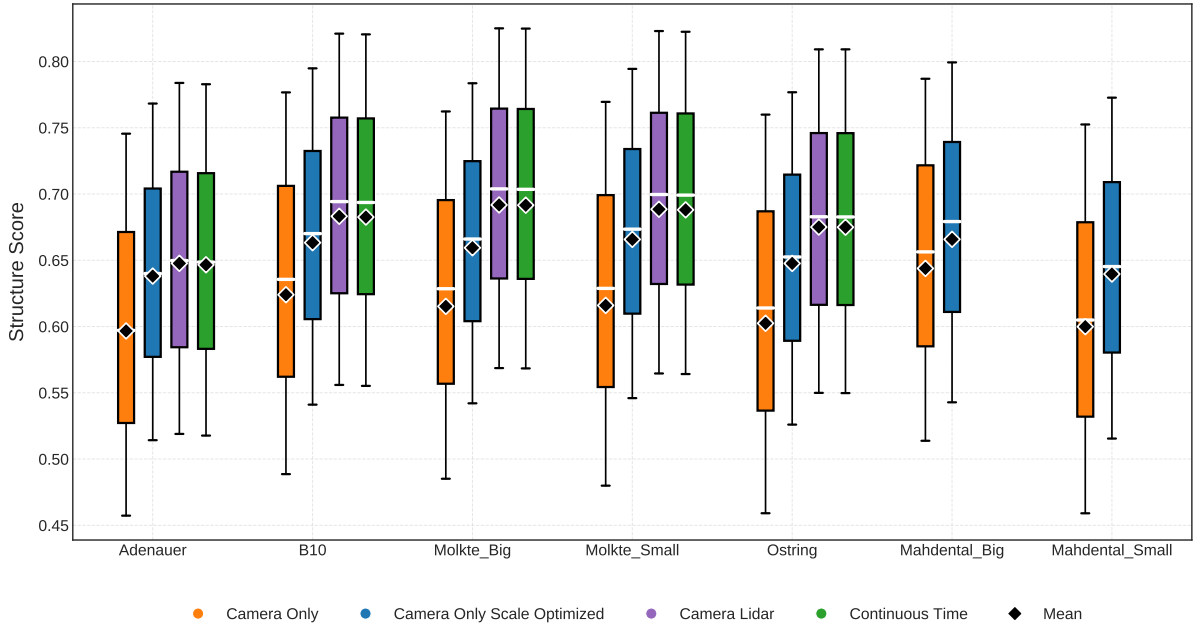


Figure 6.5: Structure scores of the accumulated LiDAR point clouds using the estimated trajectories from visual graph SLAM (orange), visual-LiDAR graph SLAM (blue), and continuous-time SLAM (green) across all seven evaluation sequences. For the box plot, the 25%, 50%, and 75% percentiles are represented by the lower, middle, and upper lines of the box, respectively. The whiskers extend to the most extreme data points not considered outliers. The black colorized diamond-shaped markers indicate the mean structure scores.

6.4 Semantic Parametric Mapping

In this section, the proposed semantic primitive parametric mapping framework is evaluated through both qualitative and quantitative assessments. For qualitative evaluation, 3D map visualization and primitive camera reprojection are employed, while the mean IoU score is used for quantitative evaluation. For the qualitative evaluation, four exemplary driving scenarios from the sequence Ostring are presented and more exemplary results are presented in A.

6.4.1 3D Map Visualization

Figure 6.6 presents exemplary visualizations of generated 3D semantic primitive maps. From these visualizations, it is evident that both elevated and road marking primitives are accurately modeled, mapped, and semantically categorized, demonstrating effective preservation of semantic information.

6.4.2 3D Primitive Reprojection

Figure 6.7 illustrates exemplary camera reprojections of 3D primitive maps. From these reprojections, it is observed that the estimated 3D primitives exhibit strong spatial consistency with the corresponding camera images. Considering the combined evaluation of 3D map visualizations and 3D primitive camera reprojections, the results qualitatively confirm the accuracy and consistency of the semantic primitive parametric mapping framework.

6.4.3 Primitive Mask mean IoU Score

To show the accuracy of the parametric modeling and mapping, the mean IoU described in Section 6.2.2 is computed for all landmark primitives. As a result of the association process, each landmark is associated with semantic masks and vehicle poses. For each landmark, the mean IoU is computed between the semantic masks and the reprojected landmark model using the camera model. The mean IoUs across all landmark primitives, classes, and sequences, are shown

in Table 6.4. For curbs, which are presented as polylines, a distance threshold 25 in pixels is used to determine the overlap between two polylines.

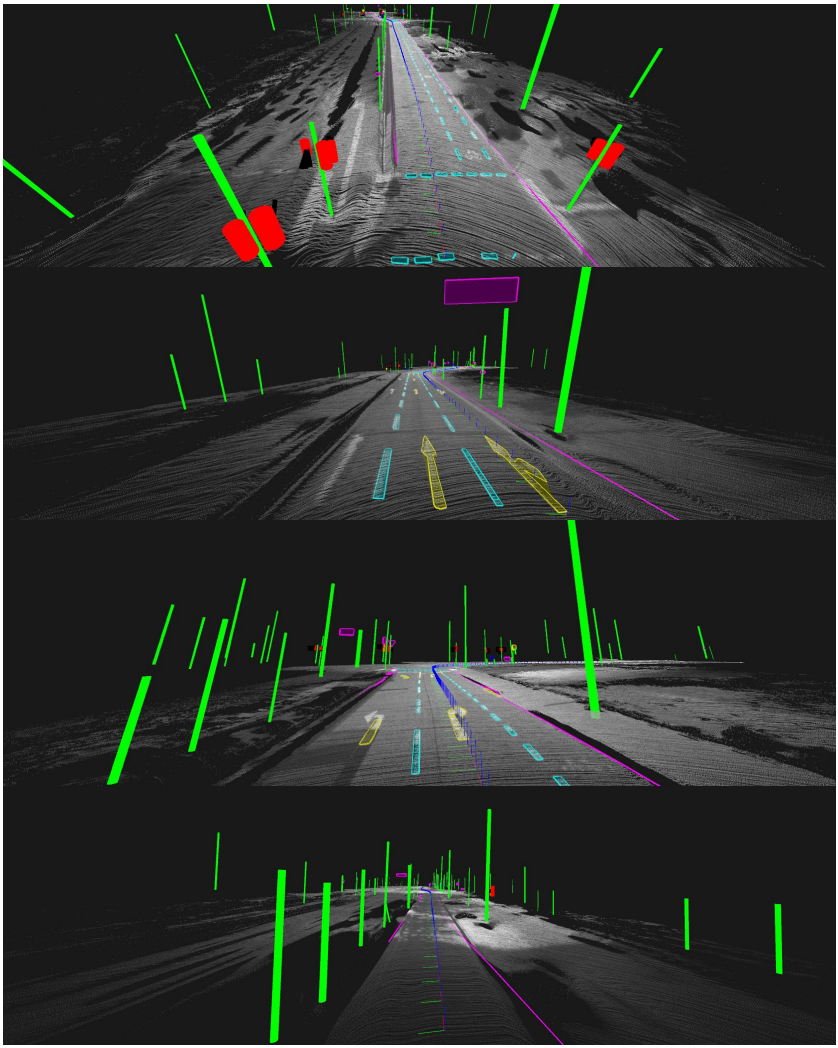


Figure 6.6: 3D primitive map examples. Poles are colorized in green, traffic lights in red with direction in black, circular traffic signs in blue, triangular, rectangular, other-shape signs, and curbs in purple, dashed lines in light blue, arrows and solid lines in yellow.

A higher mean IoU does not directly indicate better modeling and mapping accuracy, and a lower mean IoU also does not directly indicate worse modeling

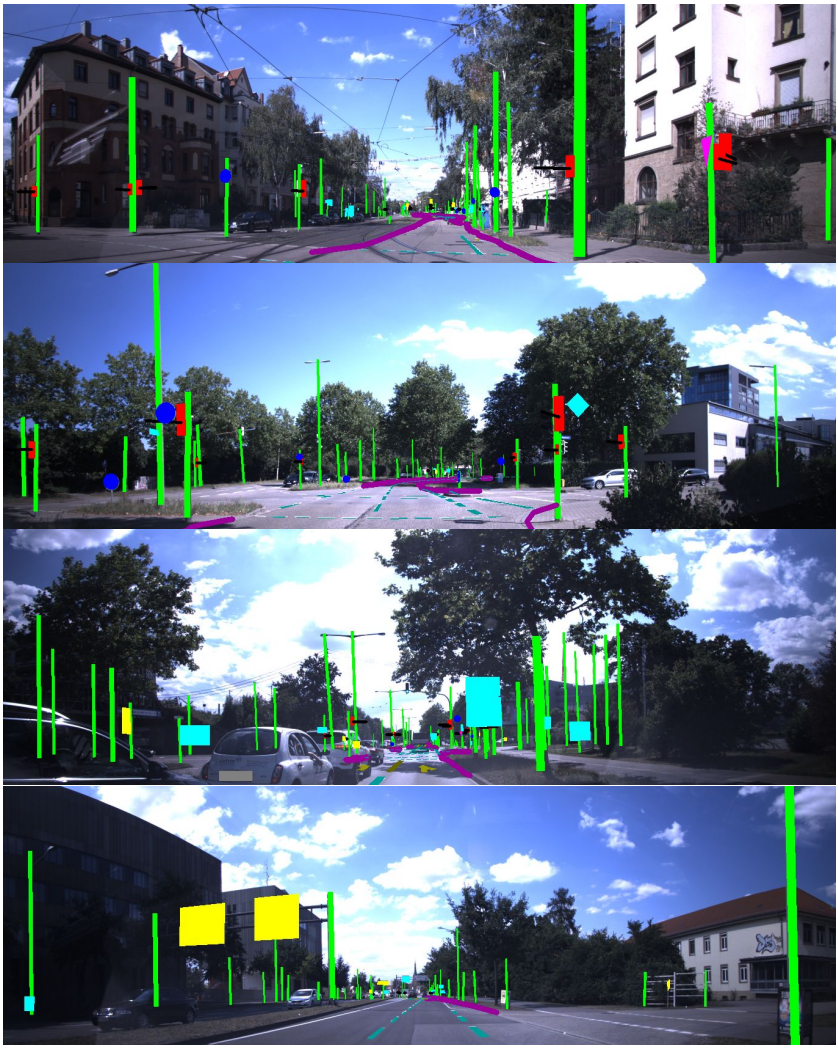


Figure 6.7: Primitive reprojection examples. Poles are in green, traffic lights in red with direction in black, circular traffic signs in blue, triangular signs and curbs in purple, rectangular signs and dashed lines in light blue, other-shape signs, arrows and solid lines in yellow.

and mapping accuracy. This is because the semantic masks are not perfect. The objects can be occluded and result in a non-perfect mask in comparison to the reprojected landmark primitive. Even though the landmark primitives are 100% accurately mapped, the mean IoU will be lower than 100%. However, from a statistical point of view, a higher mean IoU is a good indicator of the mapping accuracy quality for most cases. The landmark-based mean IoU evaluation is processed on all seven evaluation sequences, 27665 landmarks in total: 6987 poles, 1780 traffic lights, 3662 traffic signs, 12856 dashed lines, 511 pieces of solid lines, 895 arrows, and 974 pieces of curbs. As shown in Table 6.4, the mean IoUs are generally high, indicating that the semantic parametric mapping outputs align well with the corresponding camera semantic masks in terms of both pose and geometric accuracy with a mean IoU of 61.0% across all classes and sequences. Due to imperfections in the semantic masks used for mean IoU computation, the actual mapping accuracy is likely higher than indicated by the reported mean IoU values.

Similar to the non-perfect semantic masks, there are also other factors that can negatively affect the mean IoU computation. As presented in Section 5.4.1, each pole has a foot and a top point, which are initially estimated by the lowest and highest pixel points in the semantic masks projected onto the pole reference plane. According to real-world scenarios, pole top points are usually visible, but foot points are often occluded by other objects, for example, parked cars.

Table 6.4: Quantitative evaluation results using the mean IoU, reported in percentages and assessed across all seven sequences of the evaluation dataset. The metric is presented for each individual class, as a mean across classes, and as an average across all sequences. PL = Pole, TL = traffic light, TS = traffic sign, DL = dashed line, SL = solid line, AR = Arrow, CB = Curb.

Sequence	PL	TL	TS	DL	SL	AR	CB	∅
Ade	56.9	75.0	78.5	59.0	39.9	72.9	70.0	64.6
B10	51.6	74.1	81.1	50.6	42.8	66.7	62.6	61.4
Mob	52.9	74.2	79.6	45.7	34.9	63.8	61.0	58.9
Mos	55.1	74.5	80.5	49.4	26.5	62.0	68.5	59.5
Ost	55.0	74.1	81.8	53.3	51.7	70.0	68.5	64.9
Mab	47.0	70.8	74.8	47.2	35.9	70.0	66.1	58.8
Mas	45.7	71.1	73.4	47.2	36.7	69.8	68.6	58.9
∅	52.0	73.4	78.5	50.3	38.3	67.9	66.5	61.0

To estimate pole foot points as accurately as possible, the assumption is made that a pole always starts from the ground surface, based on which the foot point can be estimated by the intersection point of the pole center line and the ground surface. It helps to estimate the pole foot points more accurately and robustly, but it introduces a bias to the mean IoU computation because the estimated foot points are intended to be lower than the poles' semantic detection masks. Another factor are traffic signs with other-shape signs, which are simplified as rectangles. The real shape of these traffic signs is usually irregular and not a rectangle, which introduces a modeling error and therefore results in the computed mean IoU for the traffic sign class to be lower than the actual mapping accuracy. To clarify the effect of these two factors, a new mean IoU computation is proposed, where poles are estimated without the ground surface foot point assumption, and traffic signs with other-shape signs are excluded. The results in Table 6.5 show higher mean IoUs for both poles and traffic signs. However, this does not indicate that the ground surface foot assumption is inadequate but rather that the mean IoU computation is negatively affected by these modeling choices. An alternative approach would be to model other-shape traffic signs more accurately, similar to road marking polygons, which could further improve the mean IoU for the traffic sign class.

To further evaluate the overall robustness and generalization capability of the semantic parametric mapping framework, all mapped primitives within 150 *m* are reprojected onto every camera image and associated with their corresponding semantic instance masks. The resulting mean IoUs are presented in Table 6.6 for individual classes, the mean across classes, and the average

Table 6.5: Comparison of the mean IoUs for the landmark primitives, excluding the poles with the ground surface foot point assumption and the traffic signs with other-shape signs. The metric is presented for each individual class, as a mean across classes, and as an average across all sequences.
PL = Pole, PL no Ground = Pole without ground surface foot point assumption, TS = traffic sign, TS no Other = traffic sign without other-shape signs.

Type	Ade	B10	Mob	Mos	Ost	Mab	Mas	∅
PL	56.9	51.6	52.9	55.1	55.0	47.0	45.7	52.0
PL no Ground	59.6	55.5	56.0	57.5	57.6	50.1	48.3	**54.9**
TS	78.5	81.1	79.6	80.5	81.1	74.8	73.4	78.5
TS no Other	85.8	85.9	84.4	85.4	86.7	80.2	78.0	**83.8**

across all seven evaluation sequences. Compared with Pauls [Pau25] results for sequences Adenauer and Moltke Big (pole, traffic light, and traffic sign), the proposed framework achieves consistently improved mean IoUs. This improvement is attributed to three key advancements: (1) continuous-time SLAM framework enhancing trajectory accuracy through continuous motion representation and IMU integration, (2) multi-frame primitive association and optimization utilizing more measurements in a tightly-coupled manner, and (3) more accurate traffic sign modeling that explicitly accounts for different geometric shapes (circular, triangular, rectangular).

Table 6.6: Quantitative evaluation results using the mean IoU, reported in percentages and assessed across all seven evaluation sequences. The metric is presented for each class, as a mean across classes, and as an average across sequences. The results are compared with the partly available mean IoUs computed from the work introduced by Pauls [Pau25]. Column $\varnothing$ shows the mean IoU for the specified class and sequences Ade and Mob for comparison with Pauls [Pau25], and column $\varnothing_A$ shows the mean IoU for all sequences. Row $\varnothing_A$ shows the mean IoU for the specified sequence and classes pole, traffic light, and traffic sign for comparison with Pauls [Pau25], and row $\varnothing_B$ shows the mean IoU for the specified sequence and all classes.
PL = Pole, TL = traffic light, TS = traffic sign, DL = dashed line, SL = solid line, AR = Arrow, CB = Curb.

Sequence	**PL**	**TL**	**TS**	**DL**	**SL**	**AR**	**CB**	$\varnothing$	$\varnothing_A$
B10	16.4	28.0	29.6	7.3	11.6	10.4	8.6	24.7	16.0
Ost	18.7	23.9	29.1	10.1	13.6	13.2	13.2	23.9	17.4
Mos	20.5	26.3	23.9	7.7	8.7	14.8	13.7	23.6	16.5
Mab	21.2	25.2	29.4	9.3	8.0	7.0	10.7	25.3	15.8
Mas	16.1	28.3	27.6	8.4	11.3	7.8	15.9	24.0	16.5
Ade ([Pau25])	18.2	24.4	28.3	-	-	-	-	23.6	-
Ade	**19.5**	**24.7**	**31.1**	11.5	11.9	15.4	14.0	**25.1**	18.3
Mob ([Pau25])	18.9	25.2	28.4	-	-	-	-	24.2	-
Mob	**21.1**	**27.6**	**29.6**	10.4	8.9	8.7	10.8	**26.1**	16.7
$\varnothing$ ([Pau25])	18.6	24.8	28.4	-	-	-	-	23.9	-
$\varnothing$	**20.3**	**26.2**	**30.4**	11.0	10.4	12.1	12.4	**25.6**	17.5
$\varnothing_A$	19.1	26.3	28.6	9.2	10.6	11.0	12.4	24.7	16.7

7 Conclusion and Outlook

In this thesis, a novel multi-modal continuous-time SLAM and a semantic parametric mapping framework are introduced. The continuous-time trajectory represents motion continuously, enabling the fusion of various sensor measurements asynchronously. The semantic parametric mapping system semantically models long-term stable primitives using parametric models and tracks these primitives over time to estimate the model parameters.

7.1 Conclusion

The multi-modal continuous-time SLAM system is designed to incorporate various sensor measurements, including LiDAR, camera, IMU, and GNSS, into a bundle adjustment framework. These measurements may be asynchronous and possess different update rates, making traditional PGO methods unsuitable for data fusion, particularly when computational resources are limited. A motion trajectory is modeled using two uniform B-splines, allowing continuous-time representation, which facilitates the evaluation of motion at any given timestamp and the creation of measurement models for various types of measurements, including poses, velocities, and accelerations.

The semantic primitive parametric mapping system is developed to semantically and parametrically model and map long-term stable primitives using camera semantic images and LiDAR point clouds. For autonomous driving, long-term stable and drive-relevant primitives are crucial, such as elevated features (e.g., poles, traffic lights, traffic signs) and road marking features (e.g., dashed lines, solid lines, arrows, curbs). These features are modeled using parametric

models, such as cylinders, planes with different shapes, and polygons or polylines. It is demonstrated that the primitives are accurately and robustly modeled, with the model parameters estimated effectively by utilizing tracked semantic masks and multi-view geometry constraints.

To evaluate the proposed framework, extensive experiments were conducted on a self-collected dataset from real-world urban environments, and the results were compared with those from prior works. By integrating LiDAR point clouds into the bundle adjustment problem, trajectory estimation was significantly improved in terms of accuracy and robustness. The use of a continuous-time trajectory allows for high dynamic accuracy in motion estimation by directly fusing high-frequency IMU measurements with minimal additional computational resource consumption. Trajectory estimation was qualitatively assessed by computing the structure score of the accumulated point clouds. By modeling primitives parametrically and estimating the model parameters using multi-view geometry constraints, mapping accuracy and robustness were significantly enhanced. For the quantitative evaluation of the semantic parametric mapping, the mapped primitives were reprojected into camera images, and the consistency of the projected models with the camera images was visually assessed. For quantitative assessment, the IoU metric was computed for the semantic masks of the mapped primitives.

Besides achieving highly accurate evaluation results, the proposed framework is used as basis for downstream research topics too. A cooperation research XD-MAP presented in Bieder et al. [Bie26] is a fully self-supervised framework that uses automatically generated HD maps as an intermediate representation to transfer semantic knowledge between heterogeneous sensor modalities without requiring overlapping fields of view. It achieves a high accuracy in cross-modal semantic transfer, demonstrating the potential of the proposed framework to serve as a foundation for further research in autonomous driving.

7.2 Outlook

To further extend the proposed framework, two aspects can be considered for future work. The range of primitives can be expanded to include additional types, such as tree trunks and building facades. Furthermore, the mapped primitives could be textured using camera images, especially for traffic signs and building facades, enhancing the realism and utility of the maps for downstream applications. Another avenue for future research is to explore mapping using only camera images, particularly for scenarios where LiDAR point clouds are unavailable. A well-developed camera-only semantic primitive mapping approach would enable mapping of environments using fleet data collected from multiple vehicles, even without LiDAR. Additionally, the parametric modeling of other-shape traffic signs could be extended from the current rectangular approximation to a free-form representation, following the concave hull and polygon simplification approach already employed for road marking primitives, to achieve a more accurate geometric description of irregularly shaped signs.

Bibliography

[Aga23] Agarwal, Sameer; Mierle, Keir and Team, The Ceres Solver: Ceres Solver. Version 2.2. Oct. 2023. url: https://github.com/ceres-solver/ceres-solver (cit. on p. 25).

[Bes92] Besl, P. J. and McKay, Neil D.: "A Method for Registration of 3-D Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 14.2 (1992), pp. 239–256. doi: 10.1109/34.121791 (cit. on p. 10).

[Bib03] Biber, Peter and Straßer, Wolfgang: "The Normal Distributions Transform: A New Approach to Laser Scan Matching". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 3. Las Vegas, NV, USA, Nov. 2003, 2743–2748 vol.3. doi: 10.1109/IROS.2003.1249285 (cit. on p. 11).

[Bie23] Bieder, Frank; Hu, Haohao; Schantz, Johannes; Kirik, Oguzahn; Ries, Florian; Haueis, Martin and Stiller, Christoph: "Ein Ansatz zur automatisierten Erstellung von Trainingsdaten unter Verwendung von HD-Karten und Mehrfachbefahrungen". In: *Uni-DAS Workshop Fahrerassistenz und automatisiertes Fahren (FAS)*. FAS 2023. Berkheim, Germany: Uni-DAS, 2023, pp. 17–26 (cit. on pp. 63, 104).

[Bie26] Bieder, Frank; Königshof, H.; Hu, Haohao; Immel, Fabian; Shen, Y.; Pauls, Jan-Hendrik and Stiller, Christoph: "XD-MAP: Cross-Modal Domain Adaptation via Semantic Parametric Maps for Scalable Training Data Generation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Denver, CO, USA, 2026 (cit. on p. 120).

[Bra00] Bradski, G.: “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000). url: https://github.com/opencv/opencv (cit. on p. 89).

[Cam21] Campos, Carlos; Elvira, Richard; Gómez-Rodríguez, Juan J.; Montiel, José M. M. and Tardós, Juan D.: “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM”. In: *IEEE Transactions on Robotics (TRO)* 37.6 (2021), pp. 1874–1890. doi: 10.1109/TRO.2021.3075644 (cit. on p. 7).

[DeT17] DeTone, Daniel; Malisiewicz, Tomasz and Rabinovich, Andrew: “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Salt Lake City, UT, USA, 2017, pp. 337–33712. url: https://api.semanticscholar.org/CorpusID:4918026 (cit. on p. 8).

[Dus19] Dusmanu, Mihai; Rocco, Ignacio; Pajdla, Tomas; Pollefeys, Marc; Sivic, Josef; Torii, Akihiko and Sattler, Torsten: “D2-Net: A Trainable CNN for Joint Description and Detection of Local Features”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, 2019, pp. 8084–8093. url: https://api.semanticscholar.org/CorpusID:204241691 (cit. on p. 8).

[Eng14] Engel, J.; Schöps, T. and Cremers, D.: “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *European Conference on Computer Vision (ECCV)*. Zürich, Switzerland, Sept. 2014, pp. 834–849 (cit. on p. 9).

[Eng18] Engel, Jakob; Koltun, Vladlen and Cremers, Daniel: “DSO: Direct Sparse Odometry”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40.03 (Mar. 2018), pp. 611–625. doi: 10.1109/TPAMI.2017.2658577. url: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2017.2658577 (cit. on p. 9).

[Est96] Ester, Martin; Kriegel, Hans-Peter; Sander, Jörg and Xu, Xiaowei: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Knowledge Discovery and Data Mining (KDD)*. Portland, OR, USA, 1996, pp. 226–231. url: https://api.semanticscholar.org/CorpusID:355163 (cit. on p. 61).

[Gal15] Gallego, Guillermo and Yezzi, Anthony: "A Compact Formula for the Derivative of a 3-D Rotation in Exponential Coordinates". In: *Journal of Mathematical Imaging and Vision (JMIV)* (Mar. 2015). doi: 10.1007/s10851--014-0528-x. url: https://doi.org/10.1007/s10851--014-0528-x (cit. on p. 34).

[Gra18] Graeter, Johannes; Wilczynski, Alexander and Lauer, Martin: "LIMO: Lidar-Monocular Visual Odometry". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 7872–7879. doi: 10.1109/IROS.2018.8594394 (cit. on p. 13).

[Hal10] Hall, Brian C.: Lie Groups, Lie Algebras, and Representations: An Elementary Introduction. Vol. 222. Graduate Texts in Mathematics. New York, NY, USA: Springer, 2010, 351 S. url: https://bib-pubdb1.desy.de/record/371445 (cit. on p. 34).

[Hel62] Helgason, S.: Differential Geometry and Symmetric Spaces. 1st ed. Vol. 12. Amsterdam, Netherlands: Pure and Applied Mathematics. Elsevier Science, Jan. 1962 (cit. on p. 34).

[Hor95] Horaud, Radu and Dornaika, Fadi: "Hand-Eye Calibration". In: *The International Journal of Robotics Research (IJRR)* 14 (June 1995), pp. 195–210. doi: 10.1177/027836499501400301 (cit. on p. 103).

[Hu20] Hu, Haohao; Wang, Aoran; Sons, Marc and Lauer, Martin: "ViPNet: An End-to-End 6D Visual Camera Pose Regression Network". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Rhodes, Greece, 2020, pp. 1–7. doi: 10.1109/ITSC45102.2020.9294630 (cit. on p. 8).

[Hu21] Hu, Haohao; Sackewitz, Lukas and Lauer, Martin: “Joint Learning of Feature Detector and Descriptor for Visual SLAM”. In: *IEEE Intelligent Vehicles Symposium (IV)*. Nagoya, Japan, 2021, pp. 928–933. doi: 10.1109/IV48863.2021.9575388 (cit. on p. 8).

[Hu22] Hu, Haohao; Han, Fengze; Bieder, Frank; Pauls, Jan-Hendrik and Stiller, Christoph: “TEScalib: Targetless Extrinsic Self-Calibration of LiDAR and Stereo Camera for Automated Driving Vehicles with Uncertainty Analysis”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan, 2022, pp. 6256–6263. doi: 10.1109/IROS47612.2022.9981651 (cit. on p. 102).

[Ken15] Kendall, Alex; Grimes, Matthew Koichi and Cipolla, Roberto: “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization”. In: *IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile, 2015, pp. 2938–2946. url: https://api.semanticscholar.org/CorpusID:12888763 (cit. on p. 8).

[Kle07] Klein, Georg and Murray, David: “Parallel Tracking and Mapping for Small AR Workspaces”. In: *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. Nara, Japan, Nov. 2007, pp. 225–234 (cit. on p. 7).

[Koi21] Koide, Kenji; Yokozuka, Masashi; Oishi, Shuji and Banno, Atsuhiko: “Voxelized GICP for Fast and Accurate 3D Point Cloud Registration”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Xi’an, China, 2021, pp. 11054–11059. doi: 10.1109/ICRA48506.2021.9560835 (cit. on p. 12).

[Kro20] Kronenfeld, Barry J.; Stanislawski, Lawrence V.; Buttenfield, Barbara P. and Brockmeyer, Tyler: “Simplification of Polylines by Segment Collapse: Minimizing Areal Displacement While Preserving Area”. In: *International Journal of Cartography (IJC)* 6.1 (2020), pp. 22–46. doi: 10.1080/23729333.2019.1631535. url: https://doi.org/10.1080/23729333.2019.1631535 (cit. on pp. 94, 95).

[Küh19] Kühner, Tilman and Kümmerle, Julius: "Extrinsic Multi Sensor Calibration under Uncertainties". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Auckland, New Zealand, 2019, pp. 3921–3927. doi: 10.1109/ITSC.2019.8917319 (cit. on p. 102).

[Küm19a] Kümmerle, Julius and Kühner, Tilman: "Fast and Precise Visual Rear Axle Calibration". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Auckland, New Zealand, 2019, pp. 3942–3947. doi: 10.1109/ITSC.2019.8916987 (cit. on p. 102).

[Küm19b] Kümmerle, Julius; Sons, Marc; Poggenhans, Fabian; Kühner, Tilman; Lauer, Martin and Stiller, Christoph: "Accurate and Efficient Self-Localization on Roads using Basic Geometric Primitives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Montreal, Canada, 2019, pp. 5965–5971. doi: 10.1109/ICRA.2019.8793497 (cit. on p. 14).

[Küm20a] Kümmerle, Julius and Kühner, Tilman: "Unified Intrinsic and Extrinsic Camera and LiDAR Calibration under Uncertainties". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France, 2020, pp. 6028–6034. doi: 10.1109/ICRA40945.2020.9197496 (cit. on p. 102).

[Küm20b] Kümmerle, Julius Valentin: "Multimodal Sensor Calibration with a Spherical Calibration Target". Doctoral dissertation. PhD thesis. Karlsruhe, Germany: Karlsruher Institut für Technologie (KIT), 2020, p. 185. doi: 10.5445/IR/1000124721 (cit. on p. 102).

[Lat13] Lategahn, Henning: Mapping and Localization in Urban Environments using Cameras. Schriftenreihe des Instituts für Mess- und Regelungstechnik, Karlsruher Institut für Technologie 028. Doctoral dissertation. Karlsruhe, Germany: KIT Scientific Publishing, 2013, p. 116. doi: 10.5445/KSP/1000037227 (cit. on p. 8).

[Liu20] Liu, Lirong; Ma, Hao; Chen, Siyun; Tang, Xinming; Xie, Junfeng; Huang, Gang and Mo, Fan: "Image-Translation-Based Road Marking Extraction From Mobile Laser Point Clouds". In: *IEEE*

Access 8 (2020), pp. 64297–64309. doi: 10.1109/ACCESS.2020.2985413 (cit. on p. 16).

[Liu21] Liu, Zheng and Zhang, Fu: "BALM: Bundle Adjustment for Lidar Mapping". In: *IEEE Robotics and Automation Letters (RAL)* 6 (Apr. 2021), pp. 3184–3191. doi: 10.1109/LRA.2021.3062815 (cit. on p. 12).

[Mi21] Mi, Xiaoxin; Yang, Bisheng; Dong, Zhen; Liu, Chong; Zong, Zeliang and Yuan, Zhenchao: "A two-stage Approach for Road Marking Extraction and Modeling using MLS Point Clouds". In: *ISPRS Journal of Photogrammetry and Remote Sensing (J&RS)* 180 (2021), pp. 255–268. doi: 10.1016/j.isprsjprs.2021.07.012. url: https://www.sciencedirect.com/science/article/pii/S0924271621001970 (cit. on p. 15).

[Mor07] Moreira, Adriano J. C. and Santos, Maribel Yasmina: "Concave Hull: A k-Nearest Neighbours Approach for the Computation of the Region Occupied by a Set of Points". In: *International Conference on Computer Graphics Theory and Applications (VISAPP)*. Barcelona, Spain, 2007, pp. 61–68. url: https://api.semanticscholar.org/CorpusID:12363494 (cit. on p. 91).

[Mur15] Mur-Artal, Raul; Montiel, José M. M. and Tardós, Juan D.: "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics (TRO)* 31.5 (2015), pp. 1147–1163. doi: 10.1109/TRO.2015.2463671 (cit. on p. 7).

[Mur17] Mur-Artal, Raúl and Tardós, Juan D.: "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras". In: *IEEE Transactions on Robotics (TRO)* 33.5 (2017), pp. 1255–1262. doi: 10.1109/TRO.2017.2705103 (cit. on p. 7).

[Noc06] Nocedal, Jorge and Wright, Stephen J.: Numerical Optimization. 2nd ed. New York, NY, USA: Springer, 2006 (cit. on p. 25).

[Par07] Paris, Sylvain; Kornprobst, Pierre; Tumblin, Jack and Durand, Frédo: "A Gentle Introduction to Bilateral Filtering and its Applications". In: *ACM SIGGRAPH 2007 Courses*. SIGGRAPH

'07. San Diego, CA, USA: Association for Computing Machinery, 2007, 1–es. doi: 10 . 1145 / 1281500 . 1281602. url: https : //doi.org/10.1145/1281500.1281602 (cit. on p. 94).

[Pau21] Pauls, Jan-Hendrik; Schmidt, Benjamin and Stiller, Christoph: "Automatic Mapping of Tailored Landmark Representations for Automated Driving and Map Learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China, 2021, pp. 6725–6731. doi: 10.1109/ICRA48506.2021.9561432 (cit. on p. 14).

[Pau25] Pauls, Jan-Hendrik: "Continuous Verification and Safe Localization in Semantic High Definition Maps for Automated Driving". Doctoral dissertation. PhD thesis. Karlsruhe, Germany: Karlsruher Institut für Technologie (KIT), 2025, p. 315. doi: 10.5445/IR/1000179521 (cit. on pp. 61, 117).

[Pin09] Pink, Oliver; Moosmann, Frank and Bachmann, Alexander: "Visual Features for Vehicle Localization and Ego-Motion Estimation". In: *IEEE Intelligent Vehicles Symposium (IV)*. Xi'an, China, 2009, pp. 254–260. doi: 10 . 1109 / IVS . 2009 . 5164287 (cit. on p. 15).

[Pog18] Poggenhans, Fabian; Pauls, Jan-Hendrik; Janosovits, Johannes; Orf, Stefan; Naumann, Maximilian; Kuhnt, Florian and Mayr, Matthias: "Lanelet2: A High-Definition Map Framework for the Future of Automated Driving". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Maui, HI, USA, 2018, pp. 1672–1679. doi: 10.1109/ITSC.2018.8569929 (cit. on p. 15).

[Por19] Porzi, Lorenzo; Buló, Samuel Rota; Colovic, Aleksander and Kontschieder, Peter: "Seamless Scene Segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA, 2019, pp. 8269–8278. doi: 10.1109/CVPR.2019.00847 (cit. on pp. 14, 60, 104).

[Pro19] Prochazka, David; Prochazkova, Jana and Landa, Jaromir: "Automatic Lane Marking Extraction from Point Cloud into Polygon Map Layer". In: *European Journal of Remote Sensing (EJRS)* 52.sup1 (2019), pp. 26–39. doi: 10 . 1080 / 22797254 . 2018 . 1535837. url: https://doi.org/10.1080/22797254.2018.1535837 (cit. on p. 16).

[Red18] Redmon, Joseph and Farhadi, Ali: YOLOv3: An Incremental Improvement. Accessed: October 2025. Ithaca, NY, USA, 2018. url: https://arxiv.org/abs/1804.02767 (cit. on p. 15).

[Rou99] Rousseeuw, Peter and Driessen, Katrien: "A Fast Algorithm for the Minimum Covariance Determinant Estimator". In: *Technometrics* 41 (Aug. 1999), pp. 212–223. doi: 10.1080/00401706.1999.10485670 (cit. on p. 76).

[Sar20] Sarlin, Paul-Edouard; DeTone, Daniel; Malisiewicz, Tomasz and Rabinovich, Andrew: "SuperGlue: Learning Feature Matching with Graph Neural Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA, 2020, pp. 4937–4946 (cit. on p. 8).

[Sch13] Schreiber, Markus; Knöppel, Carsten and Franke, Uwe: "LaneLoc: Lane Marking Based Localization using Highly Accurate Maps". In: *IEEE Intelligent Vehicles Symposium (IV)*. Gold Coast, Australia, 2013, pp. 449–454. doi: 10.1109/IVS.2013.6629509 (cit. on p. 15).

[Sch14] Schreiber, Markus; Poggenhans, Fabian and Stiller, Christoph: "Detecting Symbols on Road Surface for Mapping and Localization using OCR". In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. Qingdao, China, 2014, pp. 597–602. doi: 10.1109/ITSC.2014.6957755 (cit. on p. 15).

[Sef17] Sefati, M.; Daum, M.; Sondermann, B.; Kreisköther, K. D. and Kampker, A.: "Improving Vehicle Localization using Semantic and Pole-like Landmarks". In: *IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA, 2017, pp. 13–19. doi: 10.1109/IVS.2017.7995692 (cit. on p. 14).

[Seg09] Segal, A.; Haehnel, D. and Thrun, S.: “Generalized-ICP”. In: *Proceedings of Robotics: Science and Systems (RSS)*. Seattle, WA, USA, June 2009, pp. 168–176. doi: 10.15607/RSS.2009.V.021 (cit. on p. 10).

[Sha18] Shan, Tixiao and Englot, Brendan: “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018, pp. 4758–4765. doi: 10.1109/IROS.2018.8594299 (cit. on p. 10).

[Son21] Sons, Marc: Automatische Erzeugung langzeitverfügbarer Punktmerkmalskarten zur robusten Lokalisierung mit Multi-Kamera-Systemen für automatisierte Fahrzeuge. Schriftenreihe des Instituts für Mess- und Regelungstechnik, Karlsruher Institut für Technologie 047. Doctoral dissertation. Karlsruhe, Germany: KIT Scientific Publishing, 2021, p. 151. doi: 10.5445/KSP/1000118525 (cit. on pp. 8, 63, 107).

[Sto12] Stoyanov, Todor; Magnusson, Martin; Andreasson, Henrik and Lilienthal, Achim J.: “Fast and accurate Scan Registration through Minimization of the Distance between Compact 3D NDT Representations”. In: *The International Journal of Robotics Research (IJRR)* 31.12 (2012), pp. 1377–1393. doi: 10.1177/0278364912460895. url: https://doi.org/10.1177/0278364912460895 (cit. on p. 11).

[Str15] Strauss, Tobias: “Kalibrierung von Multi-Kamera-Systemen - Kombinierte Schätzung von intrinsischem Abbildungsverhalten der einzelnen Kameras und deren relativer Lage zueinander ohne Erfordernis sich überlappender Sichtbereiche”. Doctoral dissertation. PhD thesis. Karlsruhe, Germany: Karlsruher Institut für Technologie (KIT), 2015. doi: 10.5445/IR/1000051877 (cit. on p. 102).

[Tao20] Tao, Andrew; Sapra, Karan and Catanzaro, Bryan: Hierarchical Multi-Scale Attention for Semantic Segmentation. Accessed: October 2025. Ithaca, NY, USA, 2020. url: https://arxiv.org/abs/2005.10821 (cit. on p. 63).

[Wan17] Wang, R.; Schwörer, M. and Cremers, D.: "Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras". In: *IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy, Oct. 2017, pp. 3923–3931 (cit. on p. 9).

[Wan20] Wang, Han; Wang, Chen and Xie, Lihua: "Intensity Scan Context: Coding Intensity and Geometry Relations for Loop Closure Detection". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France, 2020, pp. 2095–2101. doi: 10.1109/ICRA40945.2020.9196764 (cit. on p. 46).

[Wan21] Wang, Wei; Liu, Jun; Wang, Chenjie; Luo, Bin and Zhang, Cheng: "DV-LOAM: Direct Visual LiDAR Odometry and Mapping". In: *Remote Sensing (RS)* 13.16 (2021). doi: 10.3390/rs13163340. url: https://www.mdpi.com/2072--4292/13/16/3340 (cit. on p. 13).

[Wir21] Wirges, Sascha; Rösch, Kevin; Bieder, Frank and Stiller, Christoph: "Fast and Robust Ground Surface Estimation from LiDAR Measurements using Uniform B-Splines". In: *IEEE International Conference on Information Fusion (FUSION)*. 24th International Conference on Information Fusion. FUSION 2021. Rustenburg, South Africa: Institute of Electrical and Electronics Engineers (IEEE), 2021, pp. 1–7 (cit. on p. 41).

[Yao21] Yao, Lianbi; Qin, Changcai; Chen, Qichao and Wu, Hangbin: "Automatic Road Marking Extraction and Vectorization from Vehicle-Borne Laser Scanning Data". In: *Remote Sensing (RS)* 13.13 (2021). doi: 10.3390/rs13132612. url: https://www.mdpi.com/2072--4292/13/13/2612 (cit. on p. 16).

[Yua23] Yuan, Zikang; Wang, Qingjie; Cheng, Ken; Hao, Tianyu and Yang, Xin: "SDV-LOAM: Semi-Direct Visual-LiDAR Odometry and Mapping". In: *IEEE Transactions on Pattern Analysis and*

Machine Intelligence (TPAMI) 45.9 (2023), pp. 11203–11220. doi: 10.1109/TPAMI.2023.3262817 (cit. on p. 13).

[Zha14] Zhang, Ji and Singh, Sanjiv: "LOAM: Lidar Odometry and Mapping in Real-time". In: *Proceedings of Robotics: Science and Systems (RSS)*. Seattle, WA, USA, July 2014, pp. 109–111 (cit. on pp. 10, 41, 42).

[Zha15] Zhang, Ji and Singh, Sanjiv: "Visual-Lidar Odometry and Mapping: Low-Drift, Robust, and Fast". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA, 2015, pp. 2174–2181. doi: 10.1109/ICRA.2015.7139486 (cit. on p. 13).

[Zie14] Ziegler, Julius; Bender, Philipp; Schreiber, Markus; Lategahn, Henning; Strauss, Tobias; Stiller, Christoph; Dang, Thao; Franke, Uwe; Appenrodt, Nils; Keller, Christoph G. et al.: "Making Bertha Drive—An Autonomous Journey on a Historic Route". In: *IEEE Intelligent Transportation Systems Magazine (MITS)* 6.2 (2014), pp. 8–20. doi: 10.1109/MITS.2014.2306552 (cit. on pp. 101, 102).

List of Figures

List of Tables

A Appendix

This appendix presents comprehensive visualization results of the semantic parametric mapping system across all 7 evaluation sequences in this thesis. The purpose of this supplementary material is to provide detailed qualitative assessment of the proposed mapping framework through visual inspection of the generated semantic maps in both 3D world and 2D camera image space.

The visualization results are organized into two distinct categories: First, 3D visualizations of exemplar semantic map segments are presented for representative driving scenarios from each evaluation sequence. These 3D visualizations, shown in Figures A.1 to A.7, display the parametrically modeled semantic primitives including poles, traffic lights, traffic signs, and road markings, overlaid with reconstructed ground surface point clouds derived from the camera image pixel ground surface reconstruction

Second, 2D visualizations are provided by reprojecting the 3D semantic map elements back onto the camera image plane using the calibrated camera model and the estimated camera poses from the continuous-time trajectory. These 2D visualizations, presented in Figures A.8 to A.14, enable direct visual comparison between the parametric models and the original camera observations. The reprojection consistency serves as a qualitative validation metric, illustrating the alignment between the mapped semantic primitives and their corresponding appearances in the camera imagery.

This visualization approach facilitates the assessment of mapping accuracy, identification of potential modeling errors, and evaluation of the robustness of the primitive association and parameter estimation processes across diverse urban driving environments.

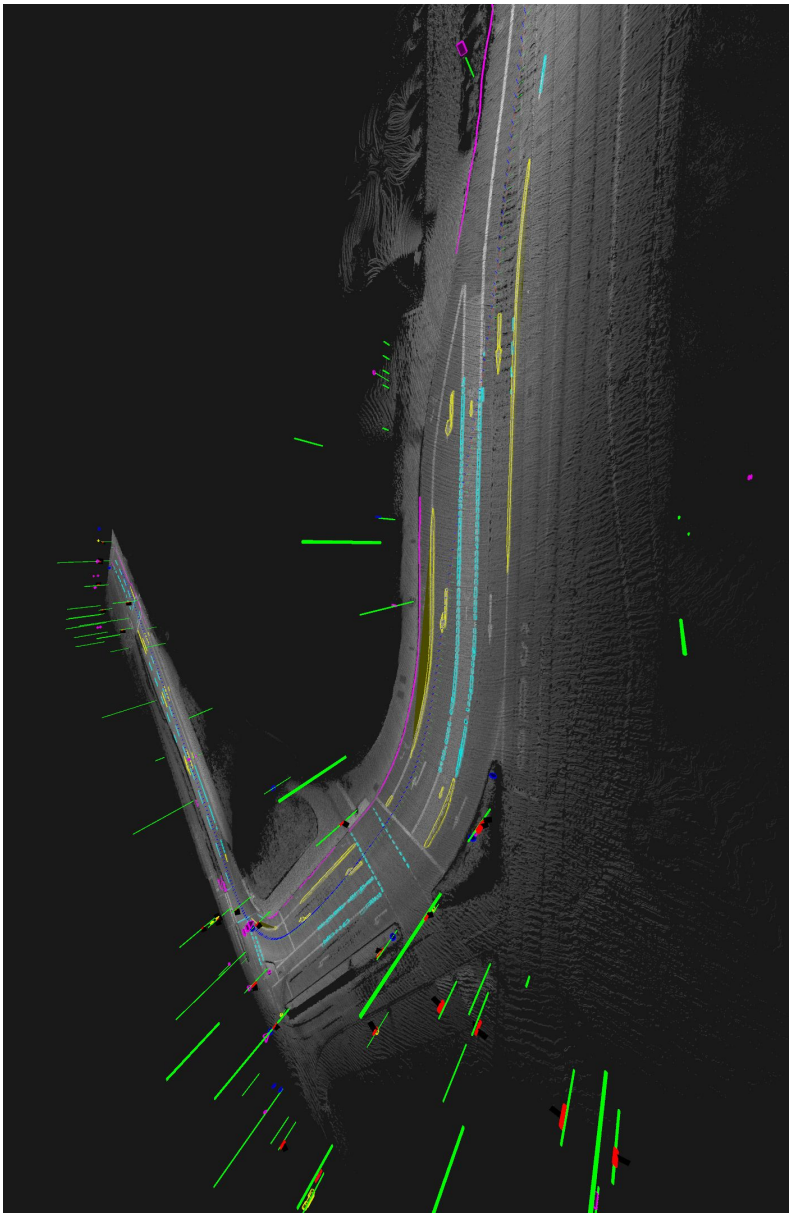


Figure A.1: 3D visualization of exemplar semantic map segment of the driving scenario in sequence Adenauer.

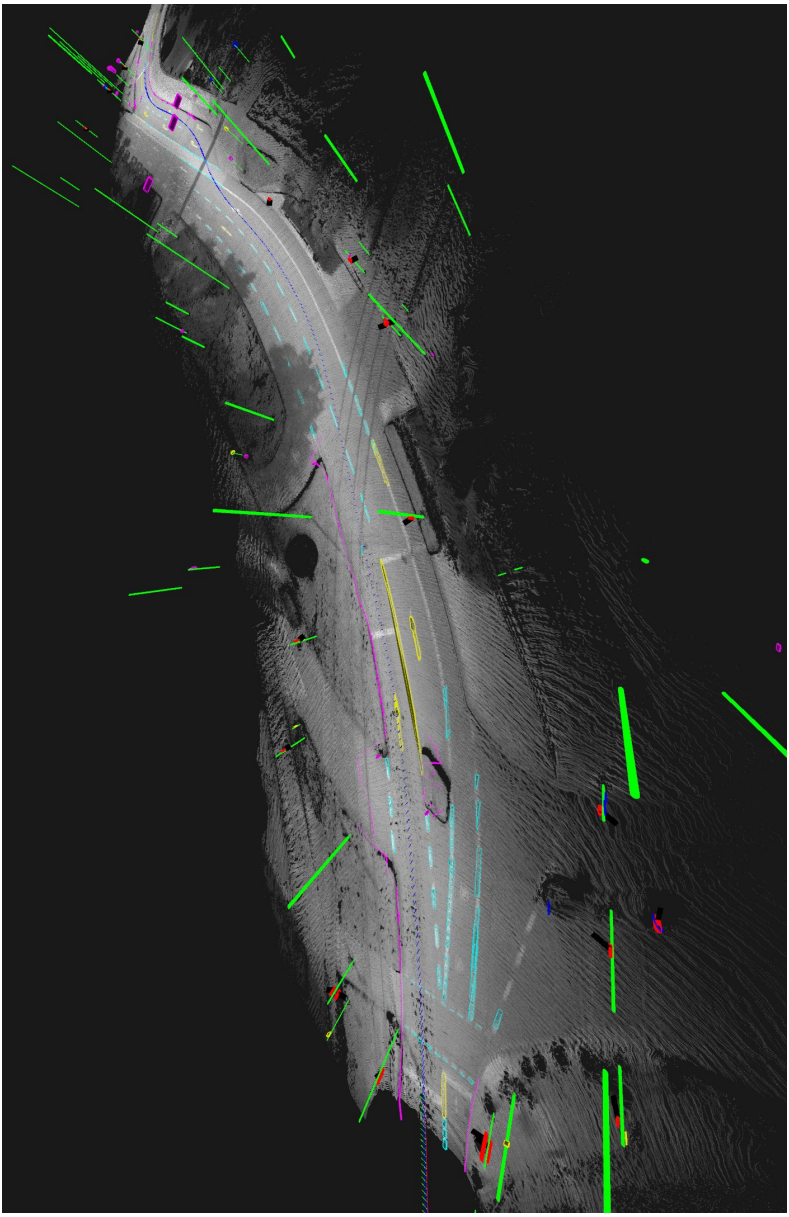


Figure A.2: 3D visualization of exemplar semantic map segment of the driving scenario in sequence B10.

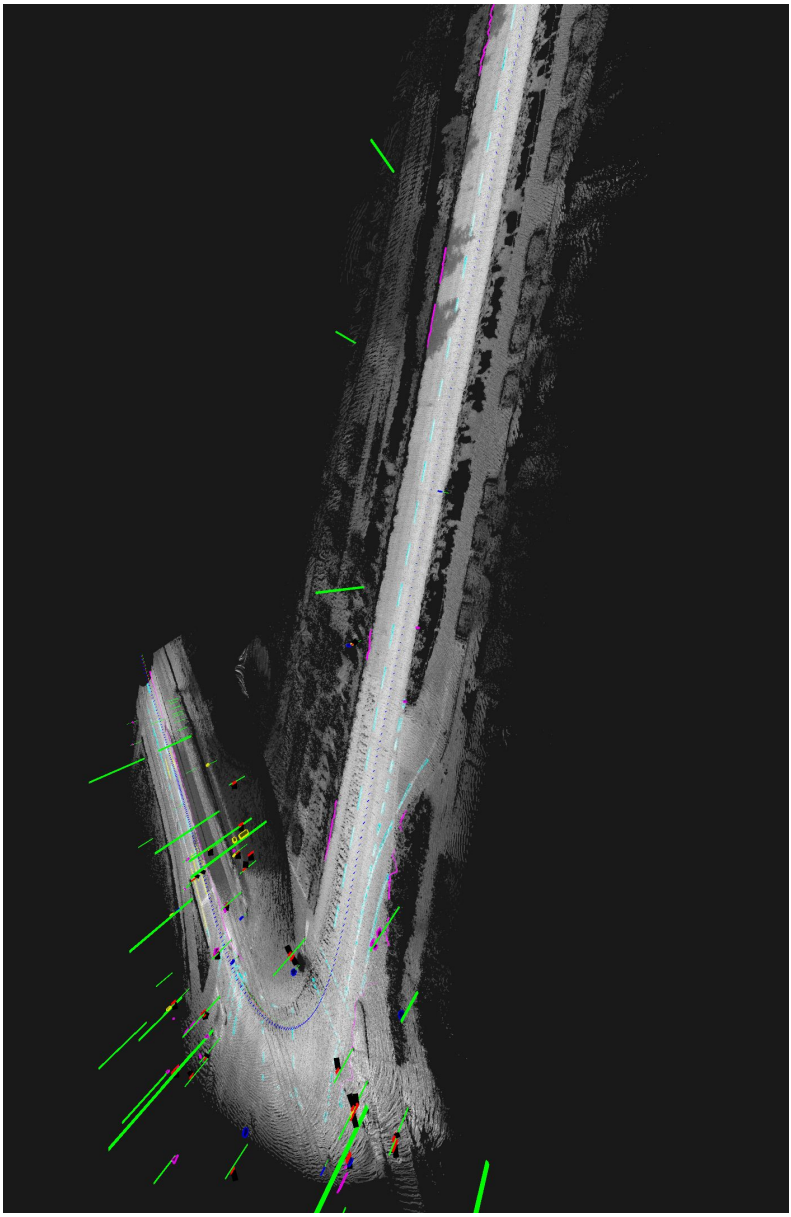


Figure A.3: 3D visualization of exemplar semantic map segment of the driving scenario in sequence Moltke Big.

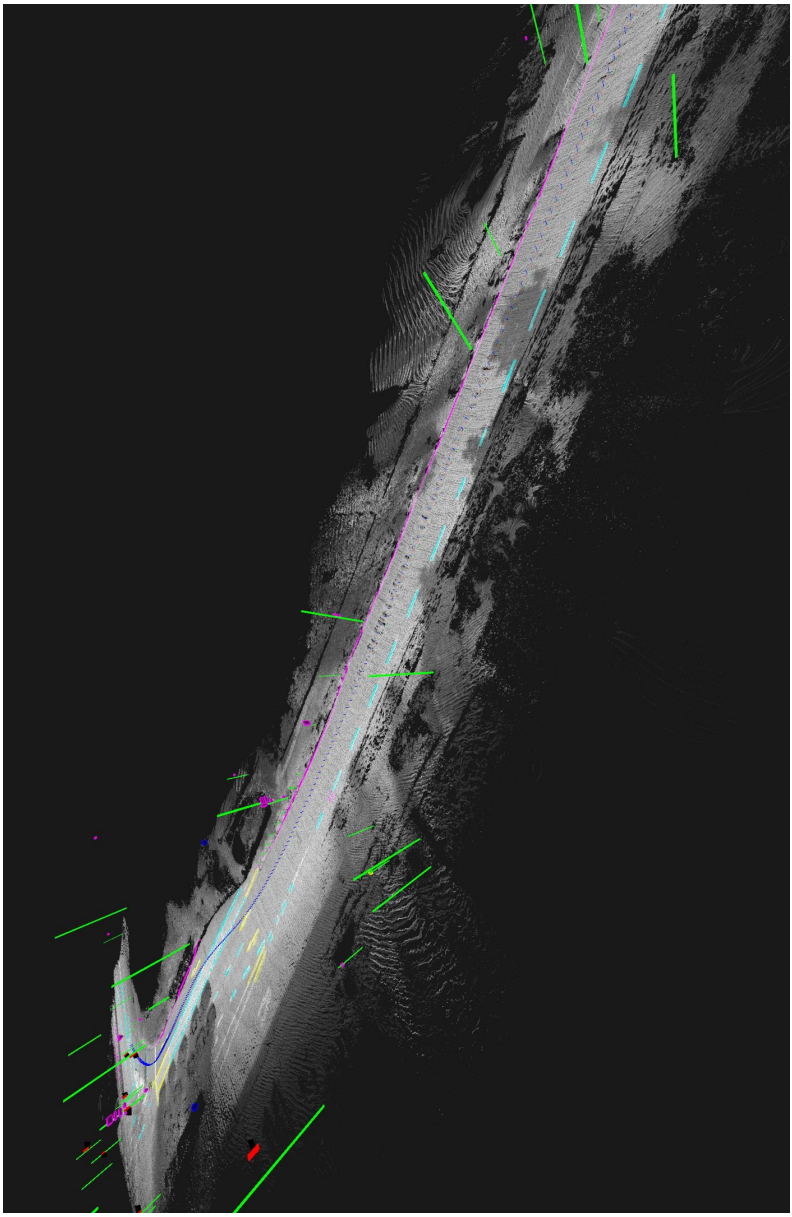


Figure A.4: 3D visualization of exemplar semantic map segment of the driving scenario in sequence Moltke Small.

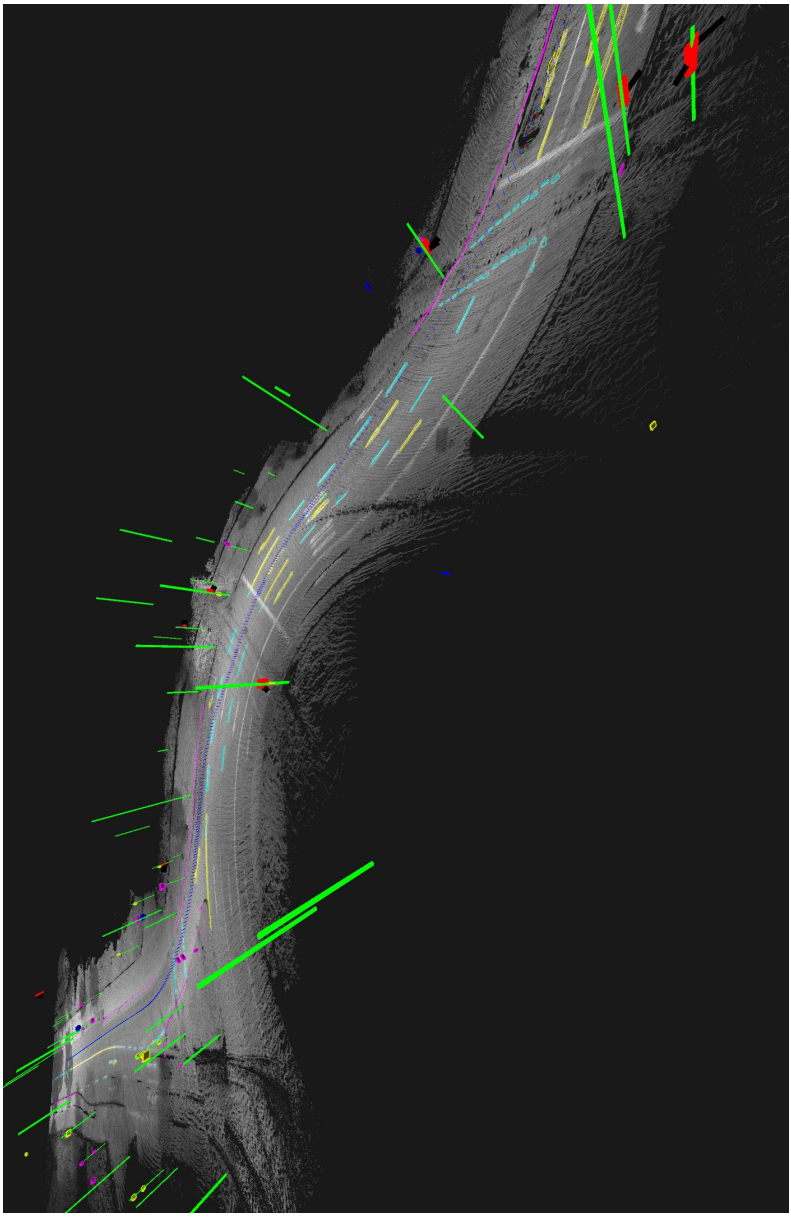


Figure A.5: 3D visualization of exemplar semantic map segment of the driving scenario in sequence Ostring.

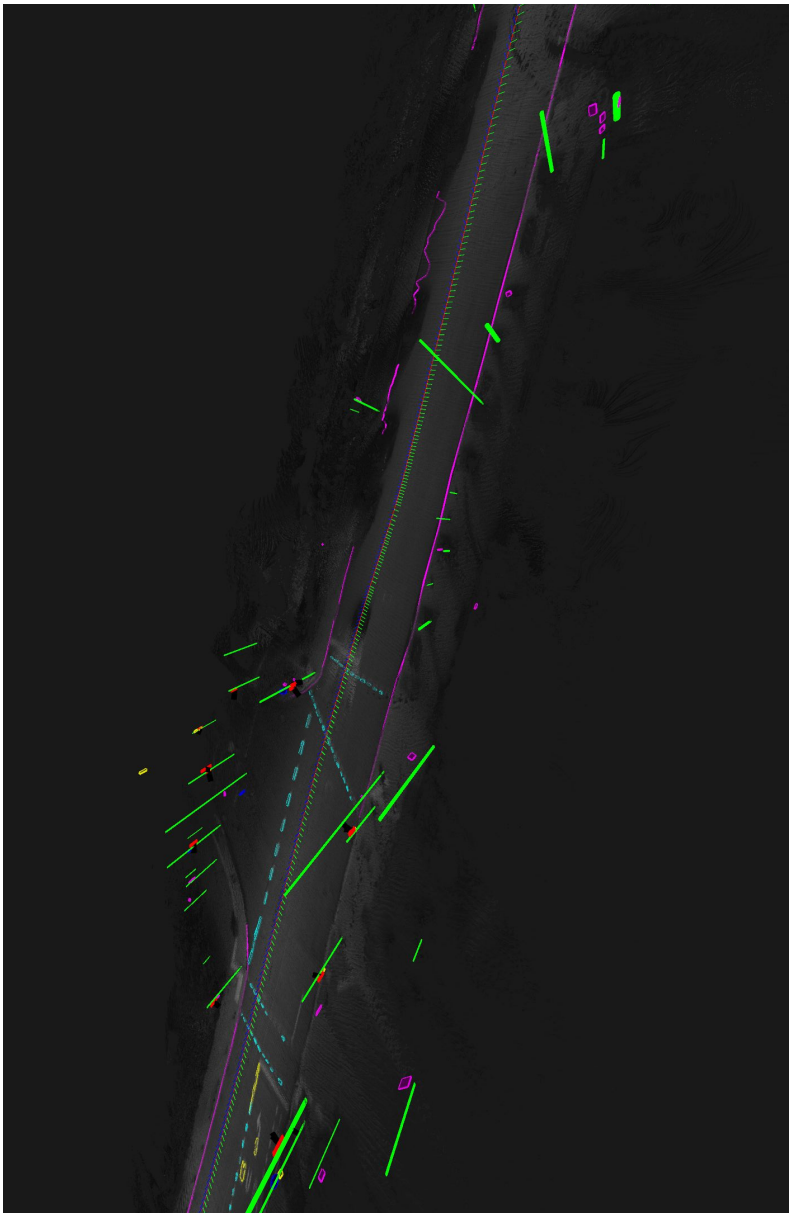


Figure A.6: 3D visualization of exemplar semantic map segment of the driving scenario in sequence Mahdental Big.

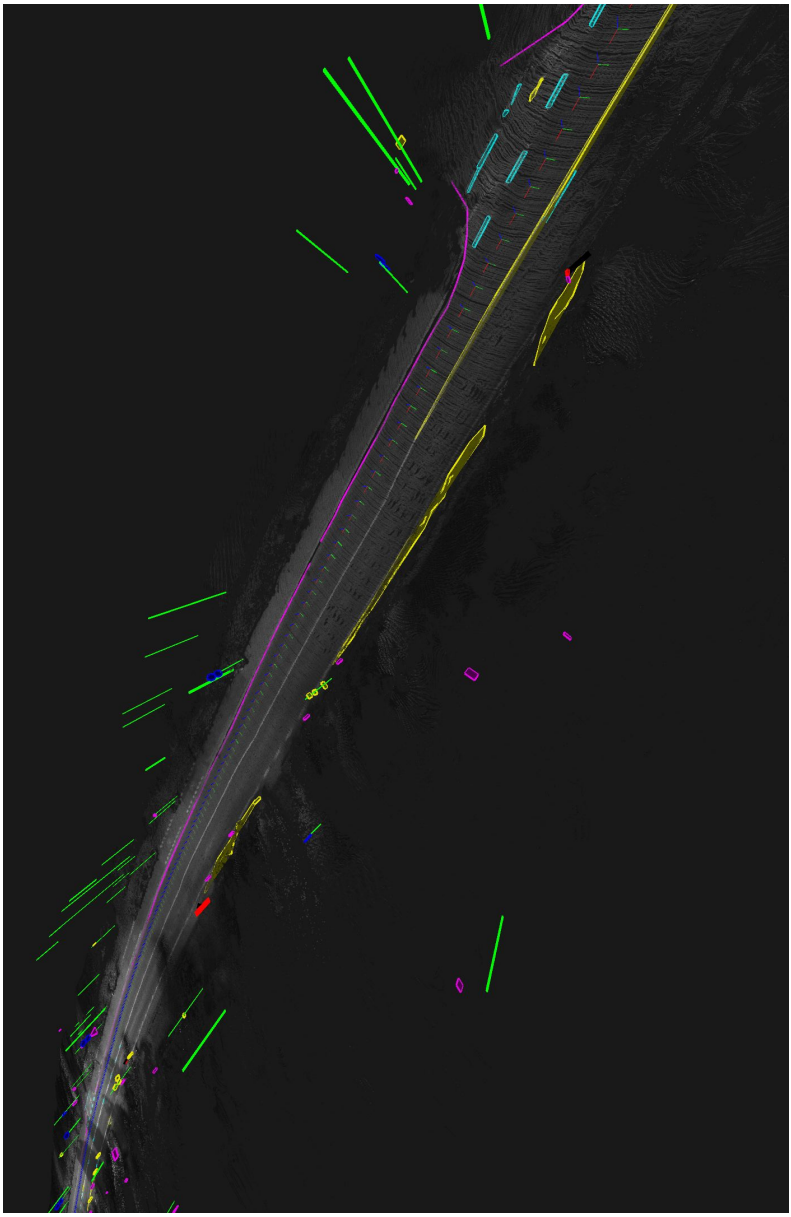


Figure A.7: 3D visualization of exemplar semantic map segment of the driving scenario in sequence Mahdental Small.

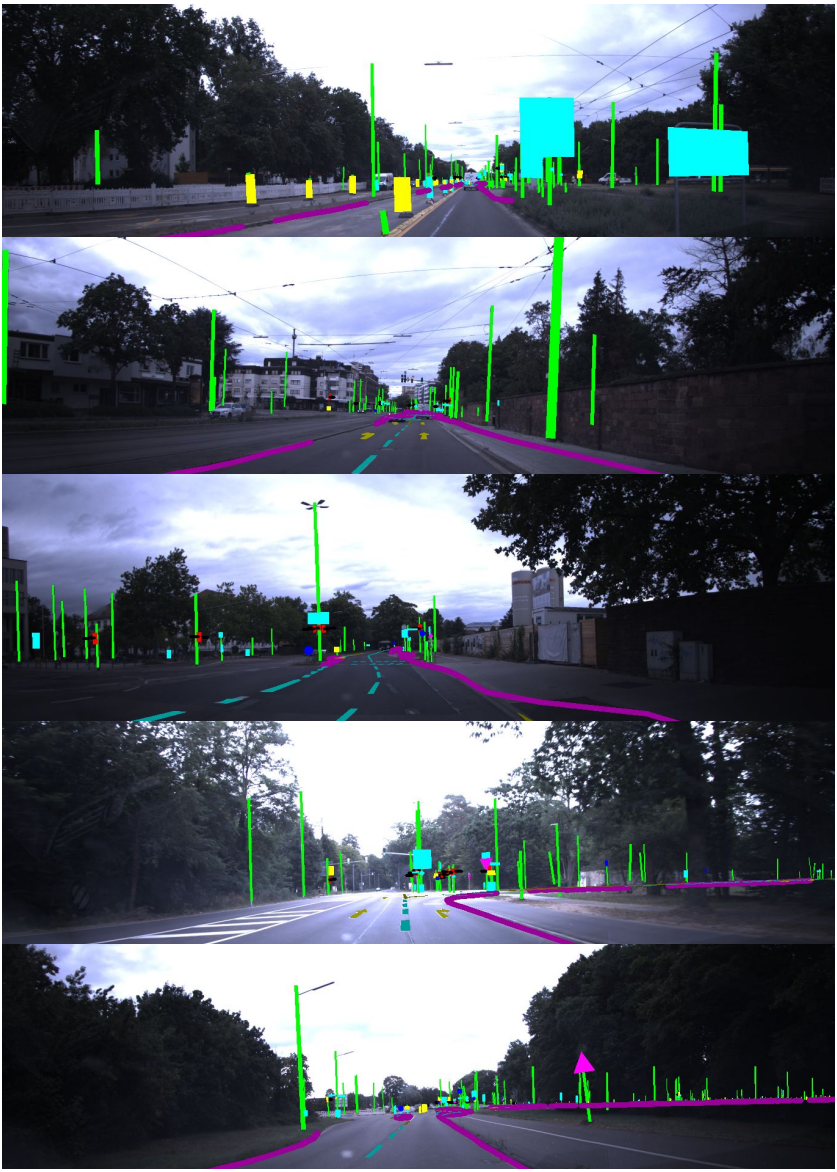


Figure A.8: 2D visualization of exemplar semantic map segment reprojected onto camera image plane of the driving scenario in sequence Adenauer.

Figure A.9: 2D visualization of exemplar semantic map segment reprojected onto camera image plane of the driving scenario in sequence B10.

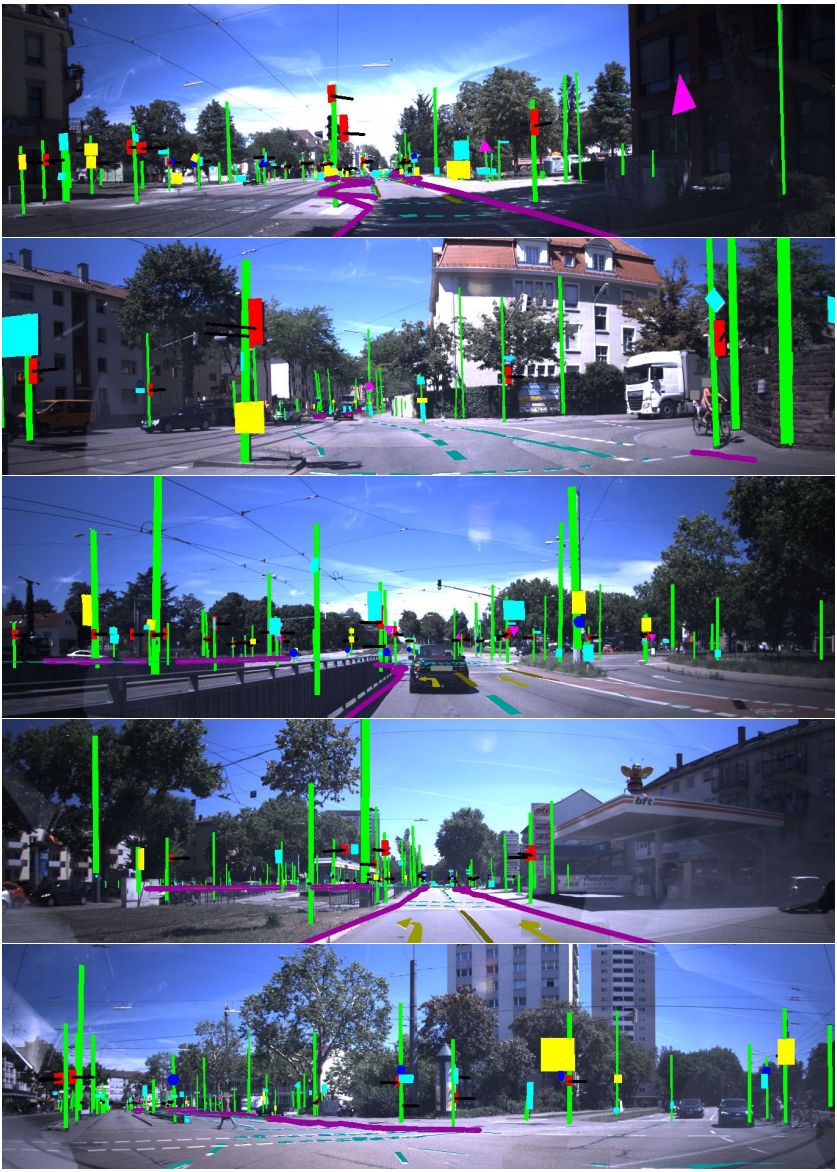


Figure A.10: 2D visualization of exemplar semantic map segment reprojected onto camera image plane of the driving scenario in sequence Moltke Big.

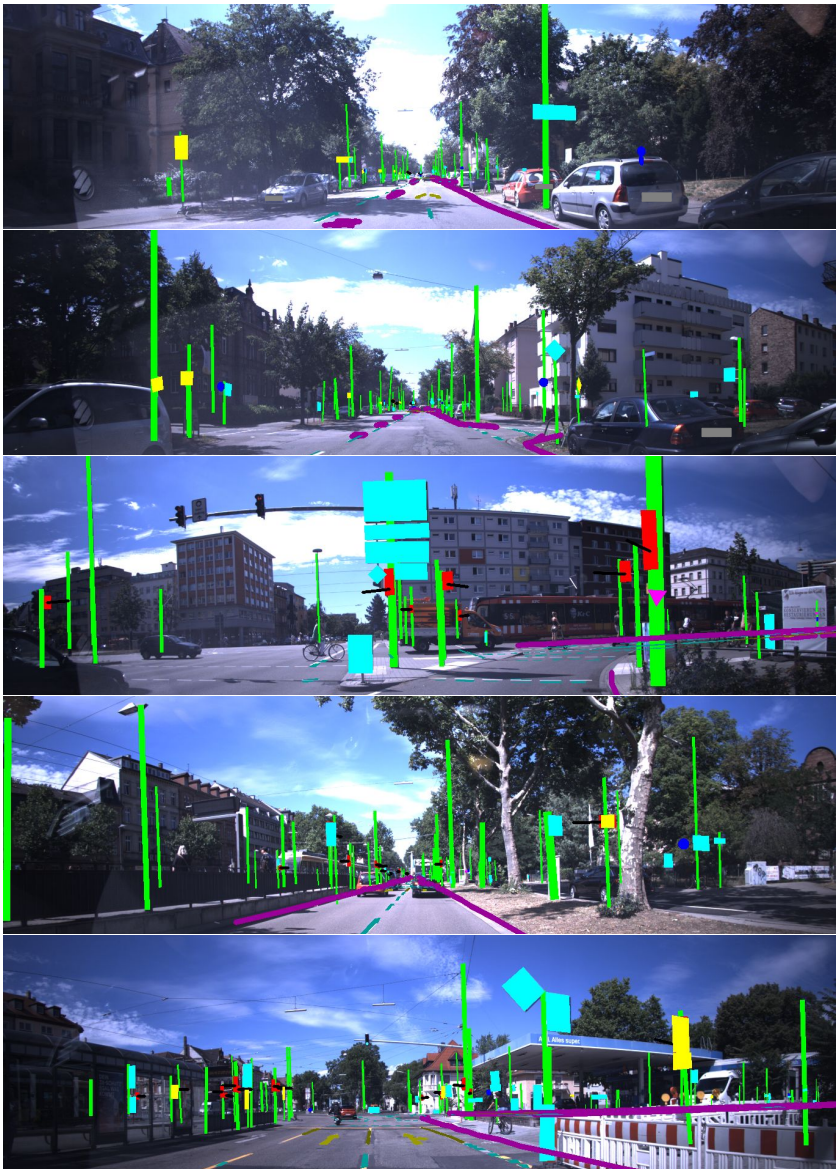


Figure A.11: 2D visualization of exemplar semantic map segment reprojected onto camera image plane of the driving scenario in sequence Moltke Small.

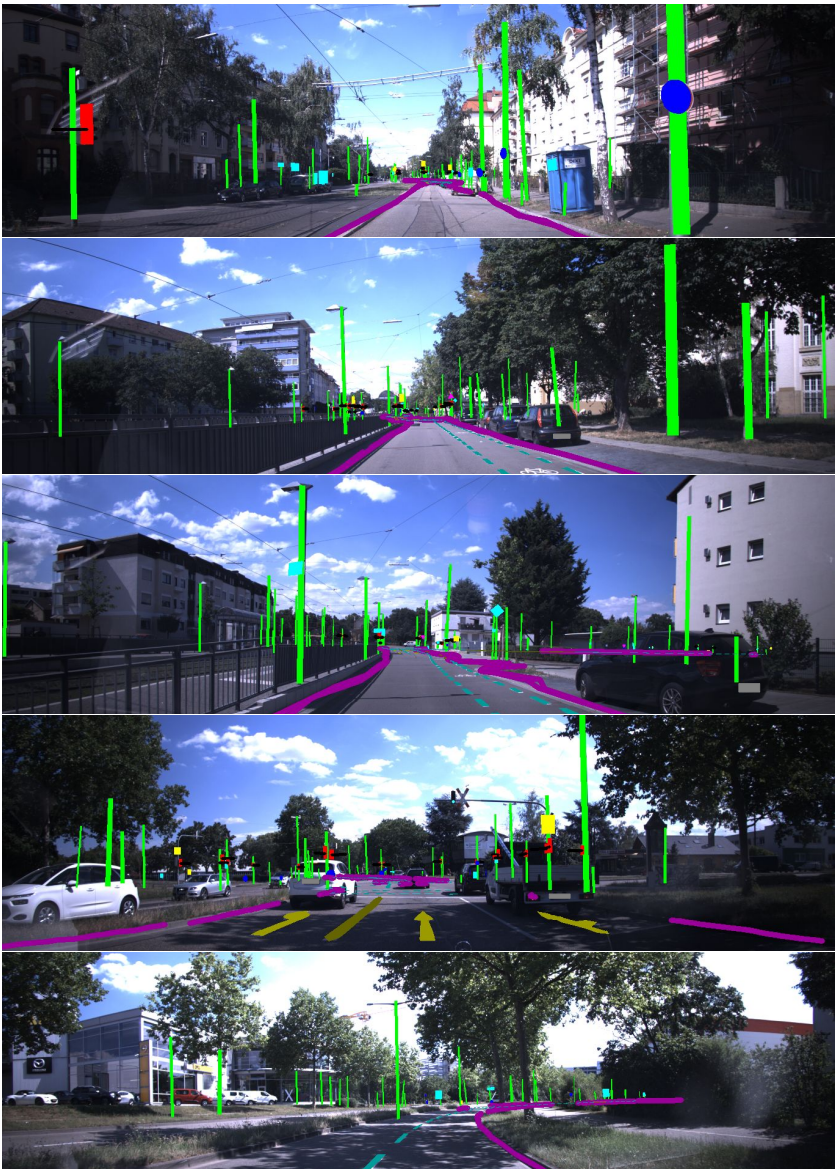


Figure A.12: 2D visualization of exemplar semantic map segment reprojected onto camera image plane of the driving scenario in sequence Ostring.

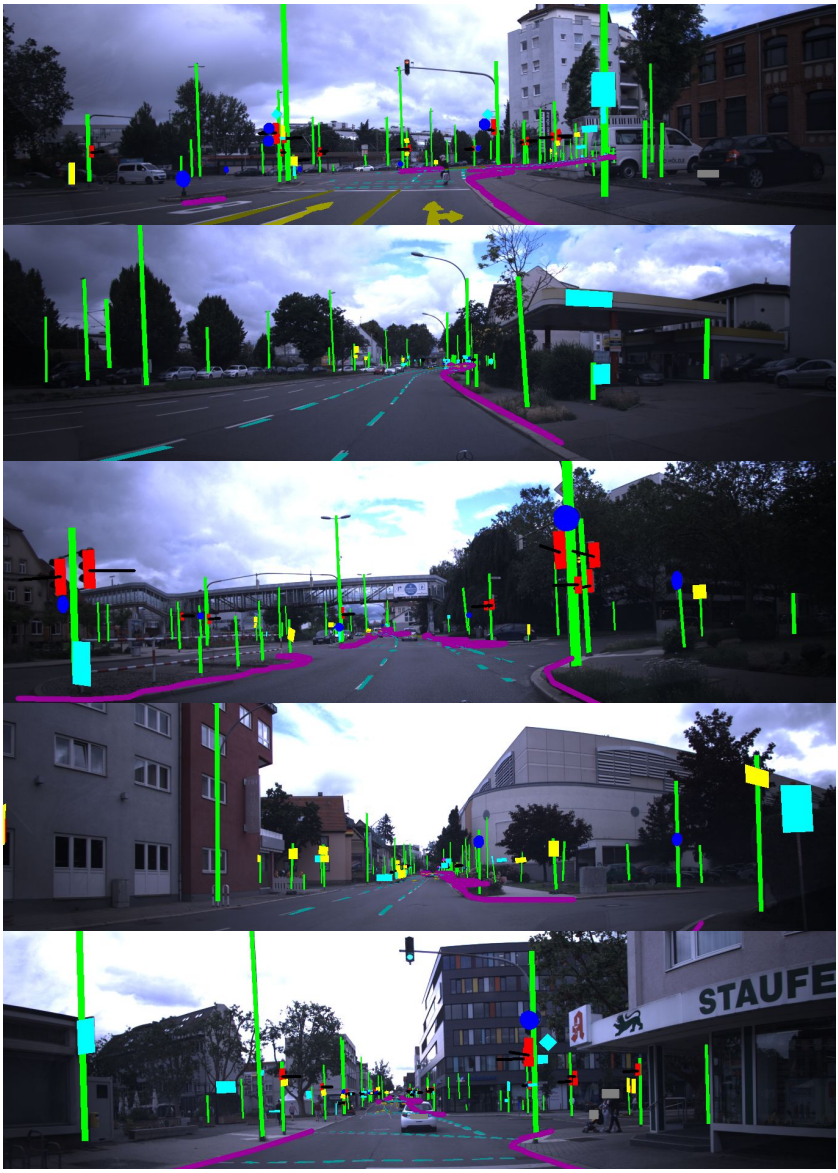


Figure A.13: 2D visualization of exemplar semantic map segment reprojected onto camera image plane of the driving scenario in sequence Mahdental Big.

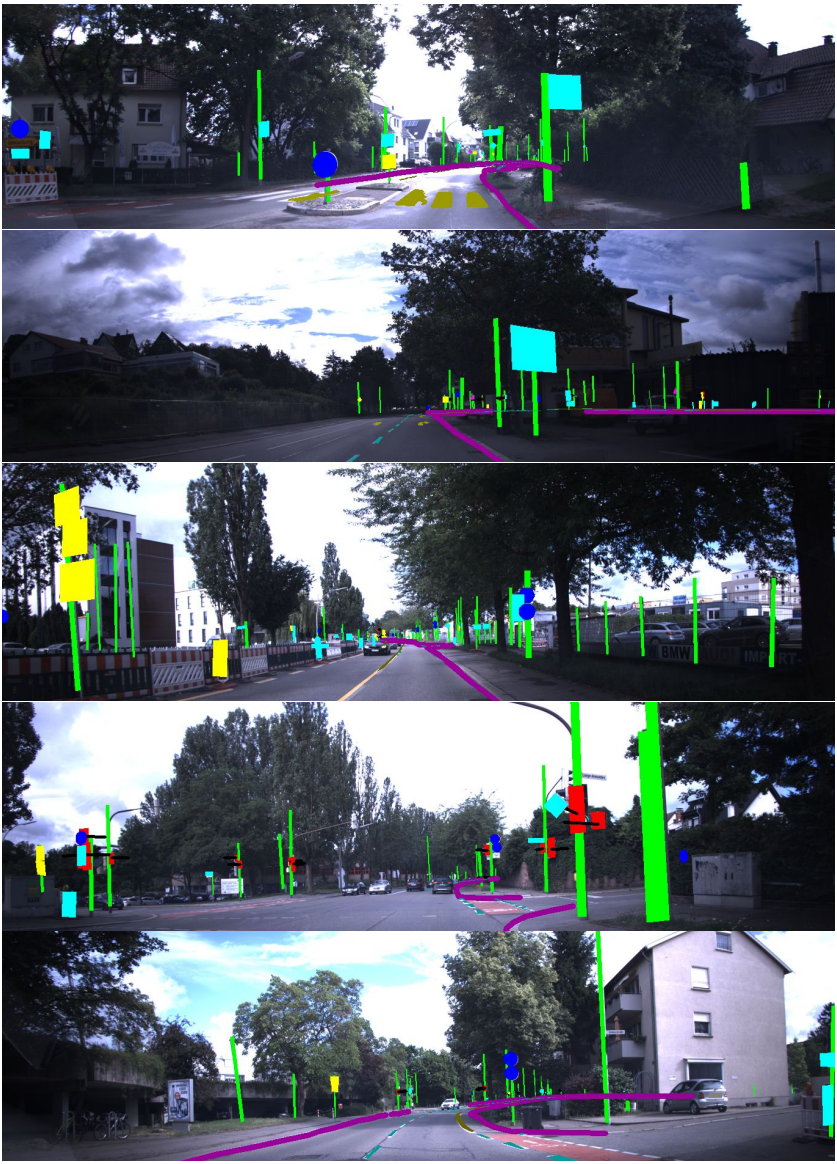


Figure A.14: 2D visualization of exemplar semantic map segment reprojected onto camera image plane of the driving scenario in sequence Mahdental Small.