Technische Universität München

TUM School of Engineering and Design

Multimodal Localization and Mapping for Autonomous Vehicles in Diverse Applications

Florian Armin Sauerbeck, M.Sc.

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung eines

Doktors der Ingenieurwissenschaften

genehmigten Dissertation.

Die Dissertation wurde am 16.08.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 05.02.2025 angenommen.

„Die weltweite Nachfrage nach Kraftfahrzeugen wird eine Million nicht überschreiten – allein schon aus Mangel an verfügbaren Chauffeuren.“

"The worldwide demand for motor vehicles will not exceed one million -– if only because of the lack of available chauffeurs."

Gottlieb Wilhelm Daimler

Acknowledgments

This dissertation was written during my work as a research assistant at the Institute of Automotive Technology at the Technical University of Munich in cooperation with the *ADAC Stiftung* and the *MCube Wiesn Shuttle* team from 2020 to 2023.

First and foremost, I want to thank my supervisor, Prof. Dr.-Ing. Markus Lienkamp. He always provided an environment full of trust and confidence to open room for outstanding research. Also, he always had an open ear for concerns, no matter what kind. I would also like to thank my second examiner, Prof. Dr. Stefan Leutenegger, for taking over the second assessment and for his valuable feedback. Furthermore, I would like to thank Prof. Dr.-Ing. Johannes Betz. Not only for taking over the chairmanship of the examination but also for supporting me the whole way during the dissertation.

From the *ADAC Stiftung*, I want to thank Thomas Heinrich for his engagement, enthusiasm, and support at the edge of research, education, and fun. The same goes for the whole IAC organization: Thank you for the outstanding memories and for having the chance to kiss the bricks! (I never thought I would thank someone in my dissertation thesis for kissing the road - but it was a great moment I will never forget!)

Big appreciation belongs to my colleagues from the chair, for great professional discussions and for becoming friends outside the work. I especially want to mention my research group AV:Perception (formerly known as Fahrdynamik and IVS), all Admin colleagues led by Dr.-Ing. Frank Diermeyer, the whole racing and EDGAR team, and the flexlab. Moreover, I extend my gratitude to my proofreaders Dr.-Ing. Alexander Wischnewski, Dominik Kulmer, Maxi Leitenstern, and Cornelius Schröder. Over and beyond, every colleague from the chair made my time so enjoyable through activities like the DokKolloq and our hiking trip to Mallorca. Thanks to you all, work never felt like work!

I want to thank all the students who contributed to this thesis and my time at the chair.

My family, especially my grandparents, parents, and siblings, deserve my deepest gratitude for supporting my way from the very beginning and enabling this dissertation.

Finally, I want to sincerely thank my girlfriend Nina for supporting me at every distance. You encouraged me to keep focus and finish this work, but in the right moments also helped me to free my mind.

Garching, May 2024

Florian Sauerbeck

Contents

List of Abbreviations

Abs-REL	Absolute Relative Error
AC@CES	Autonomous Challenge at Consumer Electronics Show Las Vegas
AD	Autonomous Driving
AECC	Automotive Edge Computing Consortium
ASAM	Association for Standardization of Automation and Measuring Systems
AV	Autonomous Vehicle
BEV	Bird's-Eye View
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoints
CaLiMO	Camera LiDAR Mapping and Odometry
CAN	Controller Area Network
CES	Consumer Electronics Show
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field
CV	Computer Vision
EDGAR	Excellent Driving GARching
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FCN	Fully Connected Network
FL	Focal Length
FLANN	Fast Library for Approximate Nearest Neighbors
FOV	Field Of View
GFTT	Good Feature To Track
GICP	Generalized Iterative Closest Point
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
HD	High Definition
IAC	Indy Autonomous Challenge
ICP	Iterative Closest Point
IMLS	Implicit Moving Least Squares
IMS	Indianapolis Motor Speedway
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KF	Kalman Filter
LiDAR	Light Detection And Ranging
LK	Lucas-Kanade

LOR	Lucas Oil Raceway
LSD	Line Segment Detector
LVMS	Las Vegas Motor Speedway
MAE	Mean Average Error
MSE	Mean Squared Error
NDT	Normal Distribution Transform
NTP	Network Time Protocol
ODD	Operational Design Domain
ORB	Oriented FAST and Rotated BRIEF
OSM	OpenStreetMap
PF	Particle Filter
PTP	Precision Time Protocol
RaDAR	Radio Detection and Ranging
RGB	Red, Green, Blue
RGB-D	Red, Green, Blue - Depth
RGB-L	Red, Green, Blue - LiDAR
RMSE	Root Mean Square Error
ROI	Region Of Interest
ROS 2	Robot Operating System 2
RPE	Relative Pose Error
RQ	Research Question
RTK	Real Time Kinematic
SAE	Society of Automotive Engineers
SAFER	Steel and Foam Energy Reduction Barrier
SD	Standard Definition
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localizaton And Mapping
SQ	Sub-Question
Std. Dev.	Standard Deviation
SURF	Speeded Up Robust Feature
TUM	Technical University of Munich
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
UTM	Universal Transverse Mercator
ViT	Visual Transformer
XML	Extensible Markup Language

Formula Symbols

Formula Symbols	Unit	Description
B	m	Stereo camera baseline
d	m	Depth / Distance to camera
f_x	mm	Focal length in x direction
f_y	mm	Focal length in y direction
K	-	Camera intrinsic matrix
P	-	Camera projection matrix
R	-	Rotation matrix
T	-	Translation matrix
Tr	-	Combined rotation and translation matrix
u	-	Horizontal image coordinate
v	-	Vertical image coordinate
σ	-	Standard deviation

1 Introduction

In 2015, the *Boston Consulting Group (BCG)* reviewed the current state and the future of autonomous driving systems [1]. They predicted that "vehicles capable of urban autopilot could be ready in 2022, paving the way for fully autonomous vehicles by 2025" in reference to *Mercedes* announcements. Retrospective, the reality looks different: Since 2022, *Mercedes* has focused on their highway pilot, neglecting urban autonomy. *Argo AI*, focusing on urban autonomous driving, was entirely closed down by *Ford* and *Volkswagen* in the same year. In 2023, *Cruise*, one of the global leaders in urban autonomous driving, was forced to halt its robotaxi operations in the United States due to safety concerns [2]. The technological and algorithmic progress has turned out to be slower than expected for several reasons. First, the complexity of the problem and the technology needed for solving was underestimated. Second, increasingly more legal and security issues arise, such as liability, data protection, privacy, or vulnerability [3]. However, the potential of autonomous driving remains enormous. It promises safer roads, fewer traffic accidents, less congestion, and more accessible mobility. This thesis will attempt to contribute to enabling robust and applicable autonomous driving, especially in the field of localization and mapping.

1.1 Motivation and Potential

When reviewing currently running public applications of Autonomous Vehicles (AVs), it becomes evident that the operation on public roads is minimal and limited. There are some running real-world applications; however, they have strict requirements for their use case and operational domain.

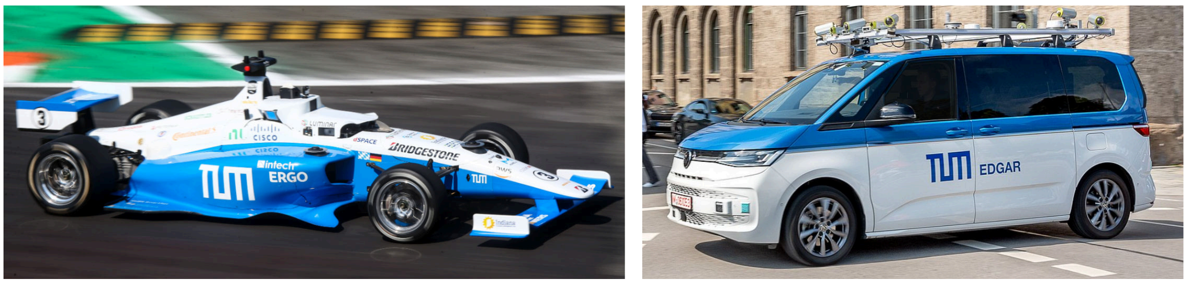
Status of Localization and Mapping

Today, it seems that the most promising approach is to start with limited and simplified applications and then steadily increase the application domain. These limited domains can be, e.g., highway pilot, cordoned-off areas, or off-road applications [4]. Data-driven approaches especially show high data dependency and insufficient capability of generalization. However, conventional algorithms also tend to overfit the specific applications for which they have been developed. Therefore, more general and better applicable algorithms are needed to advance the state of autonomous driving on public roads [4].

Precise and robust localization of the ego vehicle is a fundamental part of Autonomous Driving (AD) software [5]. Most existing approaches base the localization on using pre-built maps. However, existing approaches are overfitted to their corresponding application and use case. To further expand robots and autonomous vehicles on public roads, better generalizing algorithms are needed. Therefore, it is not sufficient to stick to public datasets for development and evaluation, but the algorithms need to be applied to real-world systems. The algorithms developed within the scope of this thesis will be applied to two vehicle platforms in entirely different applications. These will be introduced in the following section.

Vehicle Platforms and Applications

As this thesis aims to apply the algorithms to real-world use cases, developed algorithms will be evaluated with the two available research vehicle platforms: the AV-21, an autonomous formula-style race car and Excellent Driving GARching (EDGAR), a shuttle for urban traffic (Figure 1.1)



(a) IAC AV-21 (b) EDGAR research vehicle

Figure 1.1: Used vehicle platforms in this thesis.

Both of these applications come with specific challenges which have to be addressed in this thesis. In the domain of autonomous racing, localization, and mapping algorithms have to deal with poorly structured and repeating environments around race tracks, making it difficult to estimate the ego pose correctly. Additionally, high velocities cause noise in sensor data, such as motion distortion, and are responsible for large baselines between individual sensor data points. As existing approaches failed the task, this thesis will present novel algorithms to handle these challenges with the fusion of different sensor modalities, mainly Light Detection And Ranging (LiDAR) and camera (Chapter 4). It will also evaluate how these algorithms can be transferred to public roads.

For the scope of autonomous urban transport, the focus shifts towards the map creation and handling. Nowadays, only commercial providers for autonomous driving services have tools to create High Definition (HD) maps with minimal manual processing [6–8]. This thesis will provide an open-source mapping toolchain compatible with open standards for the research community. The current state of research provides several algorithms to solve partial problems, such as Simultaneous Localizaton And Mapping (SLAM). To constantly and quickly extend mapped areas, a software pipeline from raw sensor data to applicable HD maps will be developed. Existing algorithms will be enhanced and combined to create an entire open-source HD mapping pipeline. This will be applied and evaluated with the EDGAR research vehicle.

Based on all results, the discussion (Chapter 6) will give an outlook on how to create a more integrated software stack that can be applied to the two presented and future research platforms.

1.2 Autonomous Driving - Terminology and Definitions

As most robots do, AVs follow the standard sense-plan-act paradigm that leads to the three main software modules: perception, planning, and control. In the perception module, the environment is perceived through sensors, and the data is translated into an understandable format, e.g., objects, vehicles, lanes, etc. The planner uses this information to plan the vehicle's behavior, considering interaction with surrounding vehicles. The controller's task is to transform this motion plan into actuator signals and guarantee a smooth and safe driving behavior. Figure 1.2 visualizes a typical architecture for an autonomous vehicle, as also described by Pendleton et al. [5]. An overview of the current state of the art of AVs and their remaining challenges was provided by Parekh et al. [9].

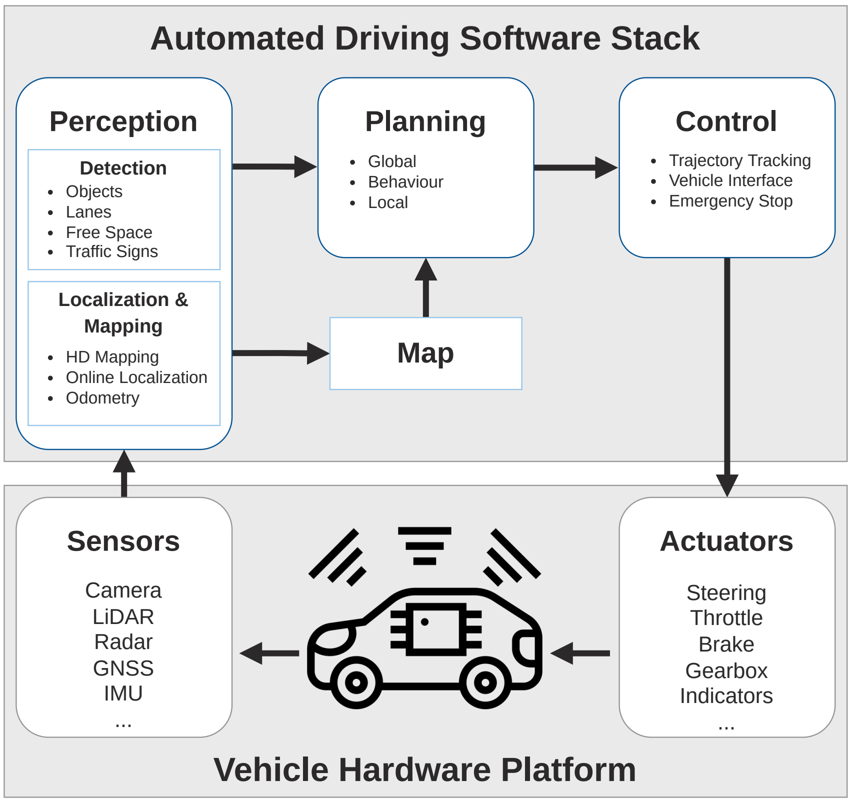


Figure 1.2: Standard architecture of an automated driving software stack.

To establish a consistent understanding, the most relevant terms for the scope of this thesis will be defined below. As far as possible, the definitions follow common usage and definitions in related work. In the case of multiple interpretations of a term in usage or literature, only the one specified here is valid in the context of this work. The terms and definitions are presented in logical order.

Automation Levels:

Initially released in 2014 and latest in 2021, the Society of Automotive Engineers (SAE) published a definition of the levels of automation for autonomous driving systems [10]. Levels 0 to 5 are defined:

- Level 0: No Driving Automation
- Level 1: Driver Assistance
- Level 2: Partial Driving Automation (Human Supervision)
- Level 3: Conditional Driving Automation (Human Fallback)
- Level 4: High Driving Automation (limited Operational Design Domain (ODD))
- Level 5: Full Driving Automation

Perception:

Perception is the process by which an autonomous system, such as an autonomous vehicle, gathers and interprets sensory information from its environment to gain an understanding of its surroundings and its own state. The main components are detection (objects, traffic signs, free space, etc.), localization, and mapping, as shown in Figure 1.2.

Simultaneous Localization and Mapping (SLAM):
SLAM is a computational technique to create maps of an unknown environment while simultaneously determining the robot's location within that environment. The foundations will be explained in Section 2.2.

Mapping:
Mapping is the process of creating a map that will be used for online localization and other driving tasks, such as path planning or motion prediction. It is usually performed offline; thus, the computation time is not critical. There is a distinction between Standard Definition (SD) maps containing basic road geometry and HD maps including details such as lanes or geometry information for localization.

HD Map:
HD maps can be assembled of different layers. As used by most current approaches and also in this thesis, in the following, this term will refer to a map consisting of two layers. A geometric layer for localization (usually a 3D point cloud map) and a vectorized road and lane map for driving tasks such as motion planning and prediction.

Localization:
Localization is the process of using online sensory data to determine the ego position within an offline-generated map. It is performed online while driving and has to run in real-time.

Odometry:
Odometry refers to the use of motion sensor data to estimate a change in position over time. In contrast to localization, no absolute position but only its change is estimated.

State Estimation:
State estimation refers to the continuous process of determining the vehicle's current position, orientation, velocity, and other relevant parameters. It provides highly frequent updates on the vehicle's current state. Therefore, different sensor modalities, e.g., Inertial Measurement Unit (IMU), Global Navigation Satellite System (GNSS), or the output from a localization module, are fused with a vehicle dynamics model. The state estimate is used by the controller to achieve the desired state.

Operational Design Domain (ODD):
ODD refers to the specific conditions and operational constraints in which an autonomous vehicle is intended to operate safely. It defines the environmental, geographical, temporal, and operational limits within which the autonomous system is designed to function. For level 5 autonomy, the system must not be limited to a specific ODD.

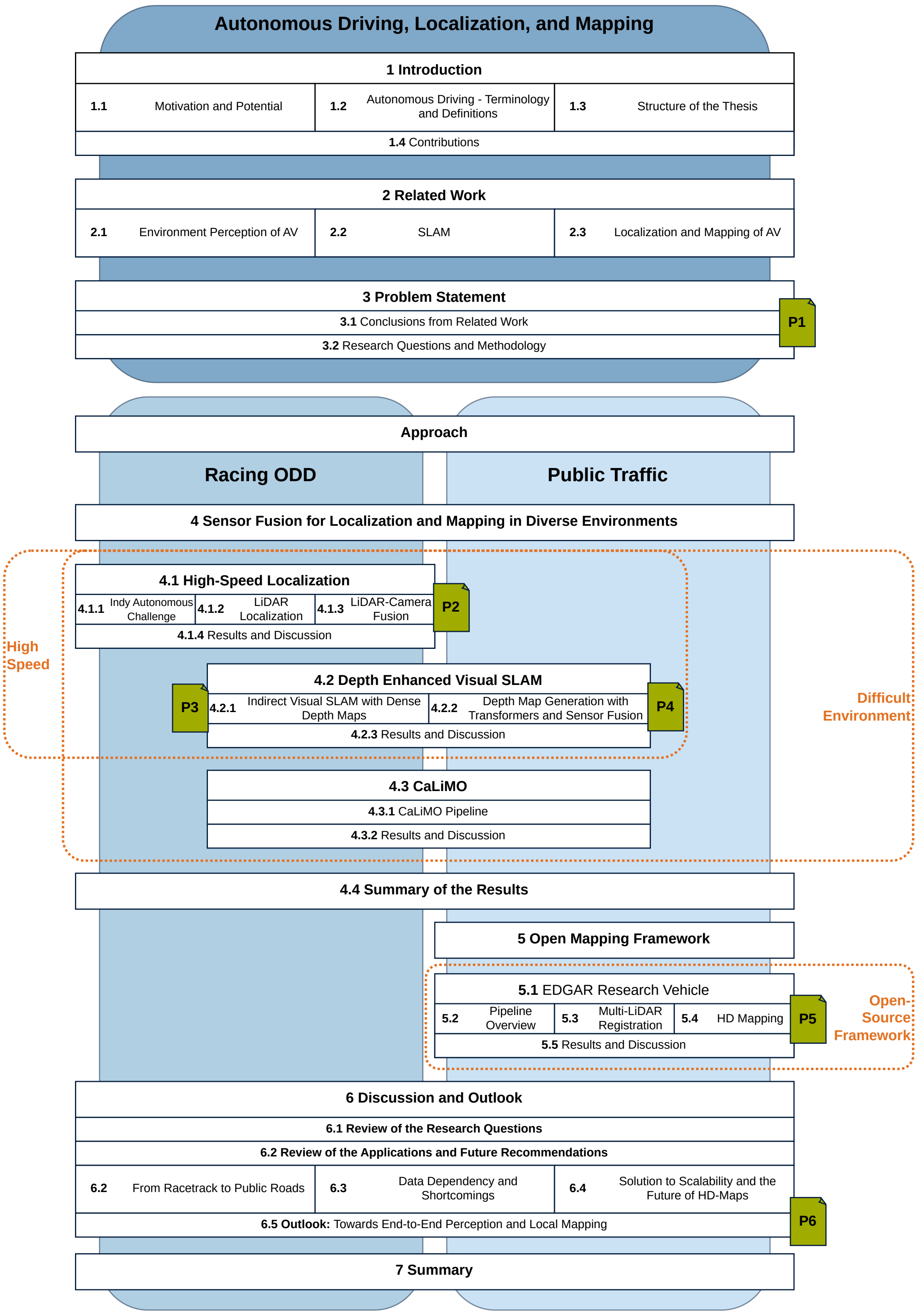
1.3 Structure of the Thesis

The structure, scope, and content of this thesis are illustrated in Figure 1.3. This figure also shows the two different applications: autonomous racing and public traffic. The corresponding publications, in which partial contents of this thesis have already been published, are shown in green.

Following the motivation for this thesis, a quick introduction of the vehicle platforms, and providing general background information in the field of AD, the contributions will be outlined in the next section (Section 1.4). This is followed by a review of related work, which provides a basis for understanding the current state of environmental perception, SLAM, and methods for precise localization and mapping (Chapter 2). The fundamentals of computer vision are introduced, and different approaches to solve the SLAM problem, using different sensor modalities, are presented. Finally, applied mapping pipelines for AVs are introduced, and their limitations are elaborated. The core of the research is articulated through a clearly defined problem statement that outlines the research gap and derives the research questions and the methodology used to address these questions (Chapter 3).

The study is divided into two operational domains: autonomous racing and public traffic. High-speed environments and race tracks induce several challenges, such as feature-poor environments, fixed specific sensor setups, and circumstances due to the high velocities. Therefore, novel sensor-fusion-based localization approaches are developed for the Indy Autonomous Challenge (IAC) (Chapter 4). These approaches are then extended to be more general and validated with public datasets for autonomous driving on public roads. The results of each algorithm are presented and discussed directly in the corresponding section. Sections based on existing publications are identified as such with an introductory sentence. The findings are then transferred to the EDGAR research vehicle. The aim is to develop an entire pipeline for offline map building and online localization in urban environments. Novel challenges are induced, such as the multi-LiDAR setup and, in particular, the usability of approaches to make them applicable in real-world applications with limited human processing effort. Based on the results from the racing domain, an open localization and mapping framework is introduced and demonstrated with the EDGAR research vehicle to address applicability of HD mapping (Chapter 5). The pipeline is validated on the Technical University of Munich (TUM) campus in Garching and in the city center of Munich.

Chapter 6 presents the high-level discussion of the thesis, all presented approaches, and their corresponding results and limitations. It examines the relationship of the approaches to the research questions. Directions for future research are defined. The discussion section extends to the dependencies of the data used, the challenges faced, and potential solutions, forecasting the future of HD mapping and the transition towards an integrated perception and local mapping system. Chapter 7 concludes the thesis with a brief but comprehensive summary.



Autonomous Driving, Localization, and Mapping
1 Introduction
1.1 Motivation and Potential
1.2 Autonomous Driving - Terminology and Definitions
1.3 Structure of the Thesis
1.4 Contributions
2 Related Work
2.1 Environment Perception of AV
2.2 SLAM
2.3 Localization and Mapping of AV
3 Problem Statement
3.1 Conclusions from Related Work
3.2 Research Questions and Methodology
P1
Approach
Racing ODD
Public Traffic
4 Sensor Fusion for Localization and Mapping in Diverse Environments
High Speed
4.1 High-Speed Localization
4.1.1 Indy Autonomous Challenge
4.1.2 LiDAR Localization
4.1.3 LiDAR-Camera Fusion
4.1.4 Results and Discussion
P2
4.2 Depth Enhanced Visual SLAM
P3
4.2.1 Indirect Visual SLAM with Dense Depth Maps
4.2.2 Depth Map Generation with Transformers and Sensor Fusion
P4
4.2.3 Results and Discussion
Difficult Environment
4.3 CaLiMO
4.3.1 CaLiMO Pipeline
4.3.2 Results and Discussion
4.4 Summary of the Results
5 Open Mapping Framework
5.1 EDGAR Research Vehicle
5.2 Pipeline Overview
5.3 Multi-LiDAR Registration
5.4 HD Mapping
P5
Open-Source Framework
5.5 Results and Discussion
6 Discussion and Outlook
6.1 Review of the Research Questions
6.2 Review of the Applications and Future Recommendations
6.2 From Racetrack to Public Roads
6.3 Data Dependency and Shortcomings
6.4 Solution to Scalability and the Future of HD-Maps
6.5 Outlook: Towards End-to-End Perception and Local Mapping
P6
7 Summary

Figure 1.3: Overview of the dissertation content and the published papers.
P1: [12], P2: [11], P3: [14], P4: [16], P5: [18], P6: [20].

1.4 Contributions

This thesis is based on peer-reviewed publications [11–14, 16, 18, 20], in which parts of this thesis have previously been published. The thesis contains contributions in the field of sensor fusion for applied localization and mapping and pixel-wise depth estimation (Chapter 4) and provides an open mapping pipeline for autonomous driving in urban regions (Chapter 5). The main contributions of this thesis can be summarized as follows.

- **A localization approach for high-speed race cars on low-feature race tracks [11–13].**
  For localization on oval high-speed race tracks, first LiDAR-based approaches are evaluated. Reasons why these approaches, which have been developed for other use cases, fail are elaborated. A novel approach using camera images for longitudinal and LiDAR point clouds for lateral localization is developed and validated on race tracks in simulation.
- **An extension of the well-known *ORB-SLAM3* to include depth measurements from LiDAR sensors [14, 15].**
  The extension allows for the input of separately generated dense depth maps in addition to RGB images from a camera. The novel *Red, Green, Blue - LiDAR (RGB-L) mode* directly reads LiDAR point clouds and uses conventional computer vision methods for fast and efficient depth upsampling. This approach has shown to be especially performant in terms of runtime and outperforms the stereo mode in specific scenarios.
- **A Visual Transformer (ViT) for pixel-wise depth estimation of camera images and sparse Radio Detection and Ranging (RaDAR) measurements [16, 17].**
  To use dense depth maps independently of expensive LiDARs, *CamRaDepth* is presented, using camera images and sparse RaDAR inputs. It combines a transformer encoder with a depth decoder and a dedicated semantic segmentation decoder branch. By this, fine details can be restored despite the sparsity of the RaDAR reflections. For an efficient training process, heavy use of transfer learning is made. The approach outperforms the state of the art and shows convincing results even if RaDAR data is unavailable during inference.
- **A concept for a novel visual-LiDAR SLAM, called *CaLiMO (Camera LiDAR Mapping and Odometry)*.**
  CaLiMO assigns depth from LiDAR to visual camera features. The graph is constructed with image flow tracking and LiDAR scan matching constraints. It shows promising results; however, future research needs to be carried out.
- **An expandable open-source offline HD mapping and online localization framework, applied to our research vehicle EDGAR [18, 19].**
  Finally, an entire pipeline is developed and presented to quickly generate HD maps of previously unmapped regions. The pipeline includes modules for 3D point cloud and semantic lanelet mapping, compatible with *Autoware*. The mapping pipeline combines open-source algorithms and represents the first available open-source mapping pipeline to create HD maps from sensor data. The pipeline and created maps are evaluated using the EDGAR research vehicle.
- **A comprehensive discussion of all topics leading to a novel perception concept [20].**
  All of the approaches mentioned above are discussed in a common chapter. From this discussion, challenges and possible solutions are presented. These are put together in a novel concept for end-to-end multi-task perception, leading to mapless autonomy for future research and applications.

All publications that have been published within the scope of this work and that this work refers to are summarized in Chapter 7.

2 Related Work

This section reviews the current state of the art in localization and mapping for AVs. Therefore, a general background on the perception of the environment by AVs is given first, including different sensor technologies, the fundamentals of machine vision, and existing datasets that can be used (Section 2.1). Subsequently, the SLAM problem is introduced, and algorithms for solving the problem with different sensor modalities and methodologies are presented (Section 2.2). The concluding part of this section shows how the presented approaches have been applied and extended. The focus is on complete pipelines from raw sensor data to HD maps and their application in AV systems (Section 2.3).

2.1 Environment Perception of Autonomous Vehicles

As environmental perception is one of the core tasks of AD, much research has already been conducted in this field. A general overview of the environmental perception of intelligent vehicles was published by Zhu et al. [21]. There are survey articles on autonomous vehicle perception systems, presenting different approaches and remaining challenges [22, 23]. In addition, the transfer from simulation to the real world has been investigated [24]. These papers focused primarily on perception algorithms, whereas other publications mainly focused on available sensor technologies and hardware [25].

2.1.1 Sensors

The following section introduces and briefly describes the most common sensors used in AVs. For additional information, surveys can be found on sensor technology [26] and sensor fusion approaches to combine sensor-specific properties [27].

Global Navigation Satellite System (GNSS):

GNSS sensors receive the current location and time from satellites in a constellation on defined orbits. To determine the global position of the sensor and provide accurate geographic coordinates, signals from at least four satellites need to be received. The most well-known standards are *NAVSTAR Global Positioning System (GPS)*, *Galileo*, *Beidou*, and *GLONASS*. The accuracy is around $1\,\mathrm{m}$ to $10\,\mathrm{m}$. To reach cm level accuracy, differential GNSS can be used with additional correction signals, such as Real Time Kinematic (RTK).

Inertial Measurement Unit (IMU):

An IMU is a sensor system that combines accelerometers and gyroscopes to measure changes in velocity and orientation. It provides information about the vehicle's linear and angular motion by tracking how it accelerates and rotates. IMUs are essential for estimating the attitude and state of a vehicle.

Inertial Navigation System (INS):

An Inertial Navigation System (INS) combines accelerometers, gyroscopes, and a computing unit to continuously estimate the position, orientation, and velocity of a moving object by dead reckoning without the need for external references.

Wheel Odometer:

A wheel odometer operates by tracking the movement made by the wheels. The vehicle's velocity and the distance traveled are then calculated using the known circumference of the wheel. It is to be mentioned that the circumference can change due to reasons like tire wear, temperature, etc.

Camera:

Cameras are optical instruments that capture visual, two-dimensional images of the surroundings. Lenses are used to adjust parameters like the Field Of View (FOV), focal length, etc., and induce distortion that has to be compensated for. Another source of distortion can be a moving camera in combination with exposure time or a rolling shutter. As cameras are passive sensors, they are highly dependent on illumination conditions. Computer vision algorithms use camera images e.g., to identify and track objects, read road signs, and interpret traffic signals. Different types of cameras exist, for example, monocular, stereo, Red, Green, Blue - Depth (RGB-D), thermal, or event cameras.

Light Detection And Ranging (LiDAR):

LiDAR sensors emit and receive laser pulses to measure distances to objects in the environment using the time-of-flight principle. They produce three-dimensional point clouds of the environment. Depending on the scan pattern of the laser, motion distortion has to be compensated for. As LiDARs are active sensors, they do not depend on external illumination; however, they can be influenced by weather conditions, such as rain, snow, or fog.

Radio Detection and Ranging (RaDAR):

RaDAR sensors emit radio waves and detect their reflections off objects. They determine the objects' distance, angle (azimuth), and relative radial velocity by analyzing the *Doppler* shift in the reflected signals. Currently, RaDARs with fixed height are the standard, with so-called “4D imaging RaDARs” becoming increasingly popular. These receive reflections from different heights. Due to their long wavelength, RaDARs are robust against different weather conditions.

2.1.2 Fundamentals of Machine Perception

In order to lay the theoretical foundation of this thesis, some fundamentals of machine perception and machine vision are established below. These mathematical relationships and sensor models form the basis for the algorithms which are presented later.

Pinhole Camera Model

The pinhole camera is a simple model to describe the relationships between the 3D environment and the 2D image captured by a camera. Figure 2.1 visualizes the model: The final image lies in the image plane (i.e., the location of the light sensor). Light rays arriving at this plane are restricted to passing through an aperture with a small pinhole, in this model, only a point. No lens is used to focus the light; thus, no distortion occurs. This limits the incoming light rays so that from each location, only the ray from one direction arrives at the sensor, allowing the image to form as shown in Figure 2.1 [28].

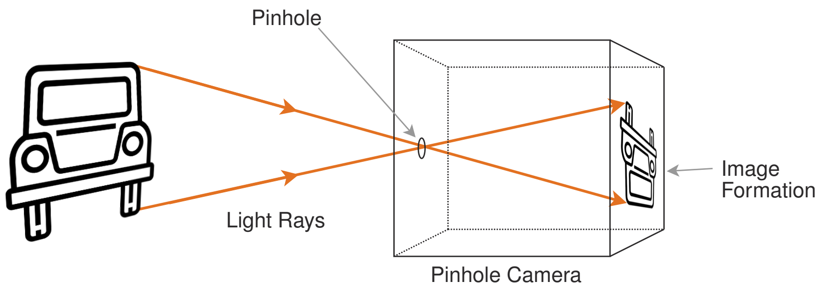


Figure 2.1: Visualization of a pinhole camera. The small opening (pinhole) blocks most of the light rays such that, from each location, only one light ray may pass through.

As real-world 3D coordinates are depicted as a 2D images, a 3D-2D transformation occurs. This projection can be described mathematically as a transformation of the real-world 3D coordinates $(x, y, z)^T$ in $\mathbb{R}^3$ to the 2D image coordinates (u, v) in $\mathbb{R}^2$ [27]. This projection is performed by the projection matrix $\mathbf{P} \in \mathbb{R}^{3x4}$. To allow a mathematical formulation of the problem, both the camera $\underline{x} = (u, v, 1)^T$ and the real-world coordinates $\underline{X} = (x, y, z, 1)^T$ are usually transformed into homogeneous coordinates.

Applying Equation 2.1 finally gives the camera coordinates of a point in the 3D world [28, 29]

$$\underline{x} = \mathbf{P}\underline{X}. \tag{2.1}$$

The matrix $\mathbf{P}$ is composed of two parts: $\mathbf{K}$ holds the intrinsic camera parameters and $[\mathbf{R}|\mathbf{T}]$ the camera rotation and translation to the world coordinate system. Equation 2.2 shows the general form of the matrix [29]

$$\mathbf{P} = \mathbf{K}\big[\mathbf{R}|\mathbf{T}\big] = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{array} \right]. \tag{2.2}$$

In the camera matrix $\mathbf{K}$, f_x and f_y describe the focal length of the camera in the x and y directions, respectively. The focal length is the distance from the lens to the image sensor and is a characteristic of every camera. Theoretically, the focal point of the image lies at the center of the image plane. The two parameters p_x and p_y allow for compensation if this does not apply to the actual camera. The parameters r_i define the rotation and t_i the translation of the camera [29]. All these parameters are measured during a process called camera calibration [30].

It should be noted that only linear corrections have been included above. In general, cameras also have non-linear distortions that may need to be corrected (e.g., introduced by lenses). In computer vision, various algorithms compensate for these errors, such as expressing several parameters as functions of pixel coordinates (u, v).

The projection described above is not invertible: The depth of a point represented by a pixel cannot be reconstructed from the 2D image alone, as no 3D depth information is contained in the 2D image. Additional devices, sensors, or algorithms are needed to reconstruct the depth information [28, 29].

LiDAR Projection

LiDAR data are typically stored as a matrix of the 3D coordinates of each received reflection. For example, in many datasets, the LiDAR point clouds are stored in a matrix $\mathbf{L} \in \mathbb{R}^{nx4}$ with n points. Each row represents one point and consists of the coordinates $\underline{X} = (x, y, z, r)^T$. x, y, and z represent the spatial 3D coordinates of the corresponding point relative to the LiDAR center. r is the reflectance value, representing the amount of light returned for each point. It depends on color, material, and surface structure. In contrast to the measured intensity value, it is scaled by the measured distance from the sensor. The points can be transformed from the LiDAR frame L into the camera image frame C to find associations between LiDAR points and pixels in the camera image plane. Therefore, the point cloud is projected to the camera origin using the projection matrix ${}^L\mathbf{P_C}$. This consists of the previously introduced camera matrix $\mathbf{P}_C$, the geometric translation matrix ${}^L\mathbf{T}_C$ and the rotation matrix ${}^L\mathbf{R}_C$ from the LiDAR to the camera. These have to be estimated via calibration. Under the assumption that the camera orientation matches the world orientation, $[\mathbf{R}|\mathbf{T}]$ (Equation 2.2) becomes the identity matrix $\mathbf{I}$ [28, 29]. Then, ${}^L\mathbf{P_C}$ can be formulated as

$$ {}^L\mathbf{P}_C = \mathbf{P}_C \left[{}^L\mathbf{R}_C | {}^L\mathbf{T}_C\right] = \begin{bmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^L\mathbf{R}_C & {}^L\mathbf{T}_C \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{2.3} $$

The homogeneous coordinate representation of a LiDAR point can be written as $\underline{\hat{X}} = (x, y, z, 1)^T$. Here, only the geometric position is considered without the reflectivity. Equation 2.4 denotes the homogeneous coordinates in the image plane of the camera

$$ \underline{\check{x}} = \begin{bmatrix} \check{x} \\ \check{y} \\ \check{z} \end{bmatrix} = {}^L\,\mathbf{P}_C\,\underline{\hat{X}}. \tag{2.4} $$

To recover the coordinates of the pixels, first $\underline{\check{x}}$ has to be normalized by $\check{z}$ (Equation 2.5) [28, 29]:

$$ \underline{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \check{x}/\check{z} \\ \check{y}/\check{z} \\ \check{z}/\check{z} \end{bmatrix}, \tag{2.5} $$

where u and v must be rounded for an association of LiDAR points with discrete pixels.

Visual Depth Perception and Estimation

In contrast to LiDAR, cameras can only sense a two-dimensional representation of the surroundings. Therefore, additional steps are required to reconstruct the depth of camera images. This reconstruction can be achieved mainly in three ways: Either through two cameras in a stereo setup, which estimate the depth by the disparity between the images, by using sparse depth inputs and ‘completing’ them, or by estimating the depth only based on 2D image inputs.

Disparity and Stereo Vision

Stereo vision is a technique for determining the distance of objects based on triangulation, similar to human binocular vision. It uses two cameras positioned at a fixed separation distance, called the baseline B, to capture images from slightly different perspectives. Figure 2.2 shows the fundamental principle of depth estimation using stereo vision.

Two cameras, Camera 1 and Camera 2, are rigidly mounted side by side with a fixed baseline B. Each camera captures an image of the same scene from its respective viewpoint. The observation of an object, for example, the car in Figure 2.2, by both cameras results in its image appearing at different lateral positions on the image planes. These are denoted by x_1 in Image 1 and x_2 in Image 2.

The horizontal shift between the object's position in the two images is the disparity δ. A computer matches the images and calculates the corresponding disparity δ for each found correspondence. The disparity is inversely proportional to the distance of the object from the camera lenses: the larger the disparity, the closer the object. This limits the range of this depth estimation approach. The object's distance or depth D can be calculated from the disparity and the focal length f of the camera. The relationship between these variables can be denoted as

$$D = \frac{fB}{\delta}. \tag{2.6}$$

Equation 2.6 is derived from the principles of triangulation, where the baseline B and the disparity form a triangle. Regarding aspect ratio, this triangle is equivalent to the triangle formed by the object and its projections in the image planes. The precise matching of point correspondences between the two images and the disparity calculation are crucial parts of stereo depth estimation.

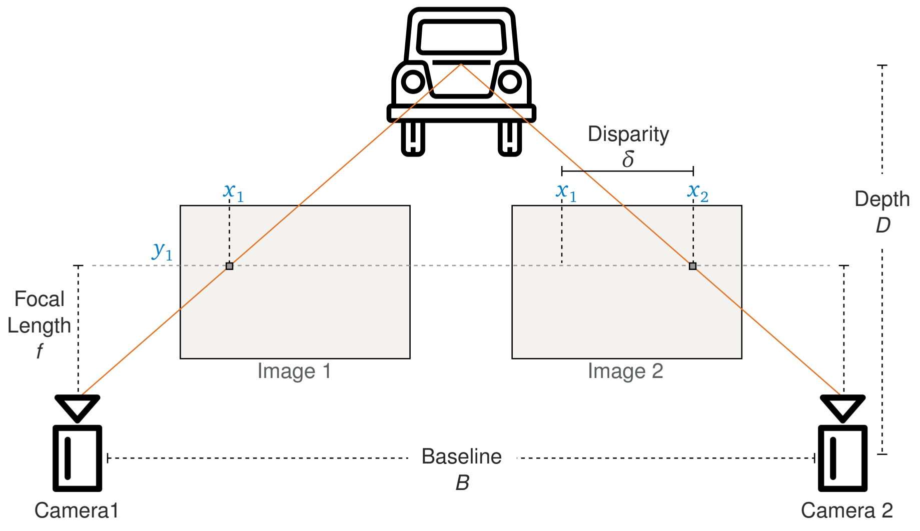


Figure 2.2: Depth estimation via disparity between images from two cameras in a stereo setup.

Depth Prediction

Another way to reconstruct depth from camera images is monocular depth prediction. In contrast to stereo view, monocular depth prediction is usually done with deep learning models, supervised or unsupervised. Zhao et al. [31] published a detailed survey on camera depth prediction.

VA-DepthNet [32] is currently the best-performing approach on the *KITTI* benchmark for camera depth prediction by adding variational constraints to a neural network. Liu et al. [33] achieved state-of-the-art results on various datasets by estimating multivariate Gaussian distributions for the depth of each pixel. *iDisc* [34] learns high-level patterns of the environment to improve supervised monocular depth prediction. Yan et al. [35] introduced *CADepth-Net*, a self-supervised approach that shows competitive results. Self-supervised approaches use the image disparity of consecutive images due to motion to supervise the training process. These are still inferior to supervised approaches in terms of depth accuracy. However, they have two main advantages: they do not need ground-truth data and can estimate the depth of finer details and structures.

Depth Completion

In addition to two-dimensional camera images, depth completion makes use of comparably sparse depth measurements, e.g., from LiDAR or RaDAR. The aim is to precisely interpolate depth information for each pixel of the corresponding camera image between the depth measurements.

The reviews by Hu et al. [36] and Xie et al. [37] present current approaches and compare their results. Most recent publications use supervised deep learning algorithms to estimate dense depth. However, some approaches use conventional Computer Vision (CV) methods [38]. *CompletionFormer* [39], *DynSPN* [40], and *SemAttNet* [41] are the best-performing approaches for depth completion validated on the *KITTI* dataset [42]. They are all supervised approaches using a depth ground truth for training. *CompletionFormer* outperforms other approaches by integrating convolutional attention into ViT blocks. The combination of a spatial propagation network with dynamic affinity matrices was presented in *DynSPN*. *SemAttNet* fuses RGB images, LiDAR data, and semantic segmentation using a learned attention mechanism [43].

2.1.3 Available Datasets

To evaluate and compare AD algorithms, common datasets are necessary. Several publicly available perception datasets have been published to advance technologies related to AD. Table 2.1 presents a curated list of prominent datasets that contribute to the research and development of HD maps, localization, and mapping for AD systems.

Table 2.1: Comparison of public AD perception datasets with a focus on localization and mapping. Note: Even if two LiDARs are used in the *Argoverse (2)* datasets, we do not consider it a multi-LiDAR setup, as there are only two similar rotating LiDARs stacked on top of each other.

Dataset	LiDAR	Camera	RaDAR	Multi-LiDAR	HD Map	Year	Note
KITTI [42]	✓	✓	✗	✗	✗	2012	
Argoverse [44]	✓	✓	✗	✗	✓	2019	
nuScenes [45]	✓	✓	✓	✗	✓	2020	HD map added belatedly
Waymo Open Dataset [46]	✓	✓	✓	✗	✗	2020	
Audi A2D2 Dataset [47]	✓	✓	✓	✓	✗	2020	5 tilted LiDARs
MulRan [48]	✓	✗	✓	✗	✗	2020	
Oxford Radar RobotCar [49]	✓	✓	✓	✓	✗	2020	
Lyft Level5 [50]	✓	✓	✗	✗	✓	2021	
PandaSet [51]	✓	✓	✓	✓	✗	2021	
3DHD [52]	✗	✗	✗	✗	✓	2022	Only HD map
Argoverse 2 [53]	✓	✓	✗	✗	✓	2023	
NTU4RadLM [54]	✓	✓	✗	✗	✗	2023	4D RaDAR + thermal
RACECAR [55]	✓	✓	✓	✓	✗	2023	High-speed (IAC)

The table indicates the sensor modalities used and whether a manually created ground-truth HD map is contained in the corresponding dataset. An additional row indicates if the recording vehicle used a multi-LiDAR setup. This makes the dataset more applicable to our research vehicles (EDGAR and the IAC racecar), as they both use a multi-LiDAR setup. In these datasets, HD map mainly stands for the vector map of the road and lane geometries. Only some datasets also provide a 3D point cloud map for localization evaluation.

The *RACECAR* dataset [55] originated from the IAC, and parts of this thesis contributed to it.

2.2 Simultaneous Localization and Mapping - SLAM

SLAM is one of the fundamental challenges in robotics and autonomous navigation systems. It is a computational process used by robots and autonomous vehicles to build a map of an unknown environment while simultaneously keeping track of its own pose within that environment. This dual-objective challenge involves the real-time acquisition of environmental data through sensors, interpreting these data to map the structure of the environment, and using the same data to deduce the vehicle's pose relative to this map. SLAM algorithms can integrate multiple data sources, including visual, LiDAR, RaDAR, and inertial, to create and continuously update a representation of the environment and the vehicle's navigation path, essential for enabling autonomous systems to move and operate independently. The following section provides an overview of the fundamentals of SLAM and existing approaches to its challenges.

2.2.1 The SLAM Process

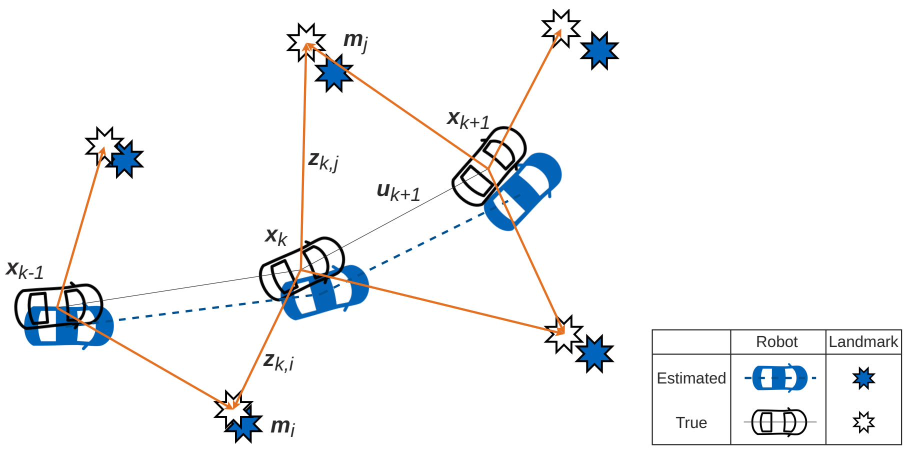


Figure 2.3: Depiction of the SLAM problem. The robot, in this case, a vehicle, needs to estimate its state x_k and the environmental landmarks m simultaneously using observations z [56]. Image adapted from [57].

The general SLAM process, applied to an AV, is depicted in Figure 2.3. The robot begins with an initial estimate of its state, x_{k-1}, and observes its environment to identify fixed landmarks, such as m_i and m_j. These observations are captured through sensor readings $z_{k,i}$ and $z_{k,j}$, which measure the relative positions of the landmarks from the robot's current location, x_k. As the robot navigates, it uses these observations to update its map, determining whether the landmarks are known (and thus updating their positions on the map) or unknown (adding them as new features to the map). The robot's next estimated position, x_{k+1}, is predicted using a motion model that incorporates the control inputs u_{k+1}, representing the robot's intended movements. The SLAM algorithm must reconcile the predicted movement with the incoming sensor data to refine the robot's position and the map's accuracy. Estimated positions of the robot and landmarks, shown in blue, may not always align with their true positions (white with black outlines). This discrepancy requires an optimization step in which the algorithm minimizes the error between the estimated and true positions. This correction is particularly crucial after a loop closure event, where the robot recognizes a previously visited location and can adjust for any drift that has occurred since then. [57–60]

Throughout the robot's travel, the SLAM algorithm continuously integrates new data, adjusts the map, and refines the estimated trajectory of the robot, ensuring that it maintains an accurate understanding of the environment and its place within it [61].

Usual SLAM algorithms can be divided into front-end and back-end [62]. The front-end processes the data from the sensors into a form suitable for analysis, while the back-end carries out the reasoning on the processed data provided by the front-end. This interaction is depicted in Figure 2.4.

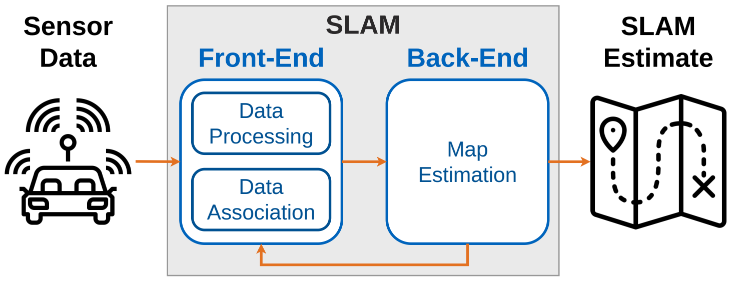


Figure 2.4: High-level overview of a SLAM implementation with front-end and back-end. Adapted from [62].

Front-End

In the SLAM front-end, the raw sensor data from consecutive frames are processed and associated to estimate the change of the sensor pose. This allows the subsequent part of the algorithm, the back-end, to reconstruct the surrounding map and the ego motion within this map.

Data Processing:
Data processing is the stage in which the sensor input is transformed into a format that can be used for trajectory estimation. For example, distinctive points or features are detected in the data. The outcome of data processing is a set of relevant data points that represent the environment in a way that is conducive to matching and map building in subsequent stages. [62]

Data Association:
Once the features or measurements have been extracted from the sensor data, the SLAM system must determine which features in the current set of observations correspond to which features in the map built simultaneously. This is critical because the correct associations are essential for maintaining an accurate and consistent map over time. Inaccurate data association can lead to errors in the estimated trajectory of the vehicle and the constructed map. [62]

Back-End

Two main methods are used to handle and further process the information from the front-end: filtering-based and graph-based approaches.

Filtering-Based Methods:
Filtering-based methods use a probabilistic approach to continuously update the map and vehicle location. As new data from the front-end arrive, filters such as a Kalman Filter (KF), Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), or Particle Filter (PF) integrate these data to revise the estimated state of the vehicle. Filtering-based SLAM considers the problem of SLAM as a state estimation problem. Data from the front-end and motion models are combined to estimate probability distributions and their associated uncertainties and provide continuous state estimates. This state comprises information about the current ego position and the surrounding map. The filter updates the state recursively to estimate the current position and map based on sensor measurements and vehicle actions. As more data are collected, the estimate is improved and refined. Filtering-based approaches excel in structured and spatially limited applications with known dynamics. Due to computational efficiency, real-time updates can be provided in such applications. [62]

Graph-Based Methods:
Graph-based methods model the SLAM problem as a graph where nodes represent the poses of landmarks or the robot, and edges represent the spatial constraints derived from sensor measurements. Such graphs are generally called pose graphs in the SLAM domain. Additional nodes are incorporated into the pose graph when new ego poses and landmarks are detected. Constraints connect the sequential nodes with information

on vehicle movement and associated landmark detections. Loop closure constraints can be added if new nodes are similar enough to previously detected ones. The goal is to optimize the pose graph and minimize errors using efficient mathematical optimization. This method is particularly effective for optimizing the entire trajectory and map globally, mainly when loop closures occur. [62]
The most famous solvers for graph-based SLAM are: *GTSAM* [63], *g2o* [64], *Ceres* [65], *iSAM* [66], *SLAM++* [67]. Nowadays, mainly graph-based SLAM methods are used as they show the best results in terms of accuracy and runtime for most applications. Moreover, this approach scales better to different sensor setups, bigger environments, etc.

Alsadik et al. [68] published a general literature survey on SLAM that included fundamental and current approaches. In addition, more specific review papers have been published on the application of SLAM in the field of AD [69, 70]. Chghaf et al. [71] conducted a literature review on SLAM for AD that additionaly focused on the implementation of multimodal approaches.

Deep Learning-Based SLAM

Recently, an increasing number of SLAM algorithms have included deep learning for different parts of algorithms, such as loop closure [72] or feature extraction and matching [73]. Other approaches make use of deep learning for the entire mapping process. In contrast to other modules such as object detection, deep learning-based SLAM approaches still lag behind the conventional SLAM approaches described above [74].

2.2.2 LiDAR SLAM

Following the introduction of the SLAM problem and general approaches in the previous sections, specific algorithms are presented below. First, LiDAR-based approaches are covered, which can be divided into registration- and feature-based.

Registration-Based:
Registration-based methods try to align consecutive point clouds to find the transformation between them. This process is also called scan matching. The most well-known scan matching algorithm is Iterative Closest Point (ICP) [75]. It iteratively refines the alignment of two point clouds by minimizing the sum of the squared distances between the single points. Generalized Iterative Closest Point (GICP) [76] uses standard point-to-point but also point-to-plane correspondences. To address the problems of ICP and GICP with outliers and initial misalignment, the Normal Distribution Transform (NDT) algorithm was proposed [77, 78].

SuMa [79] and *SuMa++* [80] are registration-based approaches. Their core includes a surfel (surface element) map, pose graph optimization, and loop closure. *IMLS*-SLAM [81] introduces a specific point cloud sampling strategy and uses Implicit Moving Least Squares (IMLS) scan-to-model matching instead of standard ICP. Even without loop closure, the approach achieves convincing results.

Recently, some new approaches have been published that tried to make more efficient use of ICP in 3D LiDAR SLAM. *KISS-ICP* [82] is an accurate and robust approach for different applications. It achieves these results by focusing on the core (point-to-point ICP and adaptive thresholding) and removing complexity rather than overfitting to a specific use case. *CT-ICP* [83] combines the continuity in the scan matching process with the discontinuity between discrete scans. This scan matching approach is integrated into a full-SLAM framework with loop closure capabilities. *Traj-LO* [84] uses a similar approach that combines point-to-plane matching with a continuous-time trajectory and is currently the best-performing LiDAR approach on the *KITTI* odometry benchmark [42]. *FAST-LIO 2* [85] is the improved version of the feature-based *FAST-LIO* [86] (presented in the next section). The authors changed the approach to direct raw scan registration and could improve computational requirements by this measure. ELO (Efficient LiDAR Odometry) [87] was developed specifically for AVs. In the 2D range image, the point clouds are segmented into ground- and non-ground points for efficient usage. Normals of non-ground points are estimated, and ground points are projected into Bird's-Eye View (BEV). Everything is combined in an ICP.

Feature-Based:

Feature-based methods first extract distinctive features (such as edges, corners, or planes) from the LiDAR data. Instead of all points in a point cloud, only these salient features are matched and aligned across successive scans. This approach can be more efficient since it processes fewer data points and can be more robust to varying densities in the point cloud. However, it may lose some detail because it disregards a portion of the data. Also, the computational effort for the feature extraction must not be neglected. Different feature detectors and descriptors have been introduced for 3D LiDAR data [88–91].

LOAM (LiDAR odometry and mapping) [92] was one of the first feature-based 3D LiDAR SLAMs. It extracts edge and planar features and estimates the robot pose by minimizing point-to-plane and point-to-edge distances. The algorithm still ranks among the best-performing approaches on the *KITTI* odometry benchmark [42]. *LOAM* paved the way for several extensions and new implementations built on it, even years later: *F-LOAM* [93] focuses on computation efficiency for real-time applications. *Lego-LOAM* [94] (lightweight and ground-optimized) adds constraints through ground segmentation and runs on embedded hardware in real-time. *LOAM Livox* [95] was specifically developed for small LiDARs with limited FOV, such as those of the company *Livox*. *M-LOAM* [96] is a multi-LiDAR SLAM algorithm based on a sliding window approach. In addition to odometry and mapping, it allows for extrinsic online calibration of the sensors by sensor-to-map matching. *Google Cartographer* [97] is a hierarchical approach working in 2D and 3D. Due to the approach with local sub-maps and global optimization, it scales well to large environments. In the *MULLS* [91] SLAM, eight different feature classes are extracted from 3D point clouds and matched by the proposed multi-metric linear least squares (MULLS) ICP. The back-end consists of hierarchical pose graph optimization for efficient computation. Many approaches use additional inertial measurements to improve accuracy. *LIO-SAM* [98] does so using a factor graph approach and also employs an IMU for motion compensation of the incoming point clouds. Additional measures, such as keyframe selection and an efficient sliding window approach, were taken to allow real-time capability. *FAST-LIO* [86] fuses LiDAR and inertial measurement with an iterated EKF. A new formula for Kalman gain computation was introduced to keep computation times low even when using a large number of measurements. *WiCFR* [99] reached convincing results among LiDAR SLAM algorithms. To robustly extract features, a point roughness evaluation based on geometric scaling was introduced. The detected features are used for the construction of motion constraints with a weighted bimodal least squares algorithm.

Deep Learning-Based:

With the increasing emergence of deep learning algorithms for various perception tasks, also deep learning-based SLAM approaches emerge. These either try to solve the SLAM problem in an end-to-end manner or substitute single modules, such as scan registration. However, as the survey shows, to this date, conventional, geometry-based approaches still outperform data-driven approaches. *DeepVCP* [100] achieves a registration accuracy comparable to the above-mentioned geometric methods. Internally, it also detects point features in the point clouds. Unlike conventional approaches, ground-truth transformations between consecutive LiDAR scans are needed for training. *PointNetVLAD* [101] is used for large-scale place recognition with deep learning-supported feature extraction. *SegMap* [102] uses a data-driven descriptor to extract features from 3D point clouds. These features are then used for map building and also were shown to be beneficial for dense reconstruction. *LO-Net* [103] is a deep LiDAR odometry, trained in an end-to-end fashion. Therefore, a geometric constraint loss is used during the training. It outperformed existing deep LiDAR odometry approaches and managed to be on par with conventional algorithms. L^3*-Net* [104] is a deep LiDAR localization, specifically designed for AVs. In contrast to the *LO-Net*, it matches sensor scans with a pre-built map, achieving cm-level accuracy. The authors argue that re-training the net for other applications is advantageous over the hand-crafted tuning of conventional algorithms.

For a more comprehensive exploration of the topic and additional insights, there are dedicated review papers about LiDAR SLAM [105–108].

2.2.3 Visual SLAM

This section presents the state of the art for camera-based visual SLAM. Similar to LiDAR SLAM, there are feature-based, also called indirect approaches and direct approaches that use the whole camera image.

Indirect/Feature-Based:

Feature-based methods focus mainly on identifying and matching striking features in camera images. There are different feature extractors and descriptors. Most of them use point features (Scale-Invariant Feature Transform (SIFT) [109], Speeded Up Robust Feature (SURF) [110], Features from Accelerated Segment Test (FAST) [111], Binary Robust Independent Elementary Features (BRIEF) [112], Oriented FAST and Rotated BRIEF (ORB) [113], Binary Robust Invariant Scalable Keypoints (BRISK) [114], Good Feature To Track (GFTT) [115], etc.). To increase robustness, line features, such as Line Segment Detector (LSD) [116], can also be used. For motion estimation, geometric reprojection constraints are leveraged. This approach can be carried out in one of two ways: through a filtering method or by employing the bundle adjustment technique.

In the filtering method, previous positions are marginalized, and the derived information is depicted as a probability distribution. This was first introduced in *MonoSLAM* [117, 118] and was solved using an EKF. *PTAM* [119, 120] implemented a bundle adjustment approach. Furthermore, keyframes were introduced, and the motion tracking and mapping threads were parallelized. *OKVIS* [121] presented a tightly coupled visual-inertial keyframe-based SLAM approach. The reprojection and temporal IMU errors were jointly optimized in a non-linear optimization problem. *ORB*-SLAM [122–124] is one of the most widely used vision-based SLAM algorithms, using ORB features for robust real-time tracking and mapping. Its parallel processes for localization, mapping, and loop closure correction enable a precise estimation of the ego trajectory and map construction, marking it as a benchmark in SLAM research. The code includes direct integrations for inertial sensors, stereo, and RGB-D setups. *OpenVSLAM* [125] is a versatile SLAM framework that supports multiple camera configurations. It employs keyframe-based mapping and loop detection to achieve accurate localization and mapping. It should be mentioned that the open-source code was taken offline due to legal concerns. *SOFT* SLAM [126] is a SLAM approach that optimizes feature-based tracking for efficiency. It integrates semantic information, optimizing data association and feature tracking in dynamic environments. Following its approach, *SOFT2* [127] builds on this foundation with improved algorithms for robustness in highly dynamic scenarios, ensuring consistent performance and reliable mapping. *SOFT2* currently holds the first place on the *KITTI* odometry benchmark [42]. Therefore, not only is the algorithm responsible, but also the camera setup was re-calibrated instead of using the default calibration files [128, 129].

Direct:

Direct Visual SLAM operates directly on pixel intensity values of raw images. This method maps changes in pixel intensity across consecutive frames to estimate vehicle movement and the structure of the environment. Consequently, these approaches use denser information than feature-based methods' sparse point features. Thus, also denser maps can be created with this approach. There are mainly two variants within Direct Visual SLAM: semi-dense, which processes selected pixels, and dense, which considers every pixel in the image. The advantage of direct visual SLAM is its ability to capture more detailed and nuanced environmental information. However, it often requires more computational power and struggles with changing lighting conditions and large baselines.

SVO-SLAM [130] (Semi-Direct Visual Odometry) is a method that combines direct and feature-based modules to achieve fast and accurate motion tracking. It directly uses pixel intensity information for odometry while maintaining a sparse feature map for robustness, optimizing for both speed and precision in real-time SLAM applications. *DTAM* [131] was one of the first published direct visual SLAM approaches. An early direct visual SLAM approach for stereo setups was *S-PTAM* [132]. *LSD*-SLAM [133] (large-scale direct monocular) is a monocular SLAM method that operates directly on the intensity values of pixels to estimate semi-dense depth maps. Using direct image alignment and efficient keyframe management, it creates large-scale, consistent maps of the environment, even on low-power devices. *DPPTAM* [134] extended the *LSD*-SLAM with color-assumptions for planar areas. *DSO*-SLAM [135] (direct sparse odometry) with the extension *Stereo-DSO* [136] focused on optimizing photometric errors over a sliding window of keyframes for direct, sparse, and model-based visual odometry. This technique follows a chosen group of pixels through multiple frames to construct a partially dense map. To make the approach more robust against changes in lightning conditions, *NID*-SLAM [137] used normalized information distance (NID) instead of photometric error minimization.

Deep Learning-Based:

Also, in visual SLAM, data-driven approaches are being published increasingly. Several approaches for visual place recognition have been developed to initialize a localization algorithm [138–142]. *Fusion++* [143] is a deep object level SLAM for densely reconstructing maps using RGB-D cameras in indoor environments. *D3VO* [144] (deep depth, deep pose, deep uncertainty) advances monocular SLAM by integrating deep learning to estimate depth, camera pose, and uncertainty. It achieves high-precision 3D mapping and robust pose estimation results. *DeepSLAM* [145] used an unsupervised learning approach with deep learning-based individual modules, including *Mapping-Net*, *Tracking-Net*, *Loop-Net*, and graph optimization. *SimVODIS* [146], and *SimVODIS++* [147] perform visual odometry, object detection, and instance segmentation simultaneously, building on top of *Mask-RCNN* [148, 149].

For further insights and perspectives on visual SLAM, the reader is directed to the extensive review articles available [150–154]. The authors of [155] applied and compared the performance of visual SLAM approaches. Some reviews specifically focus on localization and mapping for AVs [156]. The state of the art in visual SLAM will continue to evolve by integrating more sensor modalities, such as LiDARs [157]. The following section will review existing approaches in this field.

2.2.4 Sensor Fusion SLAM

The pursuit of accurate SLAM has led to the exploration of multimodal approaches to overcome sensor-specific limitations. Although camera-based systems can deliver convincing results, they often face challenges like scale drift or limited depth accuracy. Similarly, RGB-D and stereo cameras struggle with range limitations in large-scale and outdoor environments. LiDAR SLAM, in contrast, addresses these issues. However, this comes at the cost of increased hardware expenses and computational demands. Also, the limited resolution and thus attention to detail can be limiting. [71]

This section delves into multimodal SLAM, an approach that takes advantage of the strengths of various sensor modalities to improve robustness and accuracy. This section focuses on LiDAR-camera fusion approaches. The corresponding section above has already presented approaches to fuse LiDAR or camera with inertial data.

Visual-LiDAR SLAM can be divided into loosely coupled and tightly coupled strategies. As depicted in Figure 2.5, loosely coupled approaches can further be distinguished in LiDAR-enhanced visual and visual-enhanced LiDAR SLAM [158]. The following section will review the state of the art in these areas. It should be

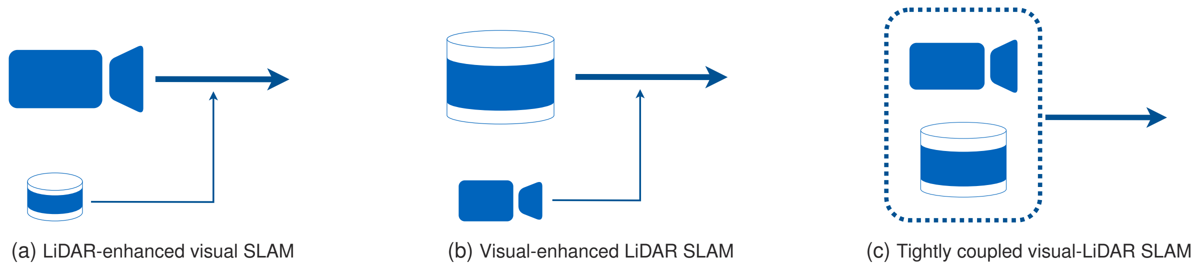

(a) LiDAR-enhanced visual SLAM
(b) Visual-enhanced LiDAR SLAM
(c) Tightly coupled visual-LiDAR SLAM

Figure 2.5: Different levels of fusion for visual-LiDAR SLAM. The first two are loosely coupled approaches.

mentioned that the distinction between these categories is not always clear-cut, as the boundaries between these approaches are often fluid and can overlap.

LiDAR-Enhanced Visual SLAM:
The most common approach in LiDAR-enhanced visual SLAM is to use interpolation between available depth measurements to retrieve the depth information of 2D image features. *DEMO* [159, 160] (depth enhanced monocular odometry) combines the depth of LiDAR with depth of triangulation through camera motion. A novel bundle adjustment algorithm uses features with available and missing depth information to estimate the vehicle's trajectory. *LIMO* [161] (LiDAR-monocular visual odometry) uses sparse LiDAR measurements, which are projected into the camera frame to reconstruct the depth of 2D features. This way, better results are achieved than with a standard feature-based visual SLAM. In [162], sparse depth measurements are projected into camera images and used for motion tracking in a direct approach. The approach showed robust results in different (large-scale) environments. *VLOAM* [163, 164] combines features with available and unavailable depth information in a high-frequency visual odometry front-end. This is combined with a low-frequency LiDAR odometry back-end for more precise and robust results. Currently, *VLOAM* ranks second on the *KITTI* odometry benchmark [42].

Visual-Enhanced LiDAR SLAM:
In visual-enhanced LiDAR SLAM, the most-used approaches are either to use visual odometry for a better initial guess for point cloud registration or to detect loop closure visually. The first was done in [165] by improving a generalized ICP algorithm with an initial transformation based on camera image features. Also, in [166], a 3D LiDAR SLAM was enhanced with a visual-inertial motion estimation front-end. [167] combines a 3D LiDAR SLAM with a visual Bag of Words [168] method based on keyframes for loop closure detection. A specific survey on LiDAR-centric multimodal SLAM approaches can be found in [169].

Tightly Coupled Visual-LiDAR SLAM:
Tightly coupled strategies for visual-LiDAR SLAM integrate both modalities into a unified optimization framework. These approaches usually use graph representations to minimize all errors in an optimization problem. They typically involve a multi-view-state formulation incorporating sensor measurements from both modalities into the state vector. Although the implementation of these strategies can be complex due to the need for raw data fusion or feature association across modalities, they tend to yield higher accuracy in practice. [71]

VIL-SLAM [170] (visual-inertial-LiDAR) was one of the first tightly coupled approaches. It combined a stereo visual front-end with LiDAR mapping and pose refinement and LiDAR-enhanced loop closure. *LIC-Fusion* [171] (LiDAR inertial camera) combines IMU data, edge features from LiDAR point clouds and FAST features from camera images in their proposed *Multi-State Constraint Kalman Filter (MSCKF)* framework. *LIC-Fusion 2.0* [172] extended the work by including plane-feature tracking based on a sliding window approach. This adaption made the algorithm more robust to uncertainty in LiDAR matching. *LVI-SAM* [173] (LiDAR-visual-inertial odometry via smoothing and mapping) incorporates two subsystems: a visual-inertial system and a LiDAR-inertial system that can work independently to handle sensor failures or jointly for improved accuracy. The data are fused in a factor graph back-end. *Superodometry* [174] is an IMU-centric approach to fuse IMU, LiDAR, and camera. It is built around a highly frequent IMU odometry, to which other available sensors (such as GNSS, LiDAR-inertial odometry, visual-inertial odometry) can provide additional constraints. The algorithm was designed in a lightweight way to account for real-time capability. R^2 *LIVE* [175] is a multi-sensor SLAM framework that integrates high-frequency filter-based odometry with low-frequency factor graph optimization. It combines camera, LiDAR, and inertial sensor data within a KF for real-time operation. The system uses a factor graph optimization to enhance the local map by refining the poses of the keyframe and the positions of the visual landmarks. R^3 *LIVE* [176] improved this work with a novel direct visual-inertial odometry front-end without the need for feature detection. In [177], visual, LiDAR, and inertial information is jointly optimized. For efficient computation, novel methods for line and planar feature extraction from 3D point clouds are presented to allow for lightweight formulation of the 3D data. All information is optimized in one single factor graph. In contrast to this, *FAST-LIVO* [178] registers raw point clouds with the map in the LiDAR-inertial odometry front-end. The map points are attached with image patches, which are then aligned with new images to minimize the direct photometric error. *TVL* SLAM [179] (tightly coupled visual-LiDAR) combines a visual odometry based on *ORB-SLAM2* [123] and a standardLiDAR odometry in a joint bundle adjustment back-end. The resulting multimodal approach significantly outperforms both visual and LiDAR SLAM. *Coco-LIC* [180] is a continuous-time LiDAR-inertial-camera odometry that integrates information from LiDAR, IMU, and camera sensors in a tight fusion strategy using non-uniform B-splines. *SDV-LOAM* [181] incorporates a semi-direct visual odometry and an adaptive sweep-to-map LiDAR odometry to effectively achieve high tracking accuracy. A sweep reconstruction is applied to increase the LiDAR odometry framerate. Additional information on multimodal, and especially LiDAR-visual(-inertial) SLAM can be found in survey papers [158, 182, 183].

2.2.5 Conclusion

It is evident that SLAM, and especially visual SLAM, is an ongoing field of research. The state of the art focuses on benchmark datasets and tries to outperform existing approaches. This leads to increasingly complicated algorithms and a limited applicability to other use cases and platforms. *KISS-ICP* [82] goes in the opposite direction. The authors prove that even if the algorithm does not rank on top of the leaderboard, the algorithm shows convincing results among several applications and datasets. Multimodal SLAM systems provide the best results in terms of accuracy. However, additional complexity is added to the algorithms, and additional requirements are set to the sensor setup. A robust and applicable approach for many real-world applications will prove superior to approaches that showed the best results on benchmark datasets and were overfitted to this data.

2.3 Localization and Mapping for Autonomous Vehicles

The last section of the related works (Chapter 2) elaborates on the state of the art in localization and mapping applied to the domain of AD. Different kinds of HD maps and methods for their generation are presented.

Maps are crucial for autonomous driving, as they provide information about the vehicle environment and thus improve situational awareness and facilitate localization [184, 185]. In contrast to sensors, maps have an unlimited range and provide information independently of external influences such as weather, daytime, etc. In addition to maps used for localization and generated via SLAM (Section 2.2), HD maps contain semantic information about the road and environment, such as lane markings, traffic lights, traffic signs, speed limits, etc. [186]. A general overview of mapping for AVs can be found in corresponding survey papers [186, 187]. The *3DHD* dataset [52] explicitly focuses on HD map usage in AD.

2.3.1 Map Formats

In general, maps for AVs can be classified into SD maps and HD maps. SD maps provide basic road geometry and topology sufficient for, e.g., navigation systems in human-driven vehicles. HD maps offer a more detailed layer of information, including lane markings, traffic signals, and curb details, which are essential for precise localization and decision-making of AVs. The granularity of HD maps allows higher accuracy and reliability in complex driving scenarios where autonomous vehicles must make nuanced navigation decisions. The following will focus on HD maps, which are used in current AD software stacks.

HD Map Structure

There are several different definitions of HD maps. The Automotive Edge Computing Consortium (AECC) defines HD maps as the topology of a road network with high precision and resolution. To allow the map to be used for precise tasks, such as localization, the term “high precision” refers to an accuracy of $10\,\mathrm{cm}$ to $20\,\mathrm{cm}$ [185, 188, 189].

To break HD maps down into their layers, there are mainly two approaches: The AECC differentiates the layers by the time interval in which they change [190]. This model is visualized in Figure 2.6. The four layers of this model are the following.

- The **Permanent Static Layer** contains information that changes within days or longer, e.g., the road structure or traffic signals.
- The **Transient Static Layer** includes elements with a lifespan of hours, such as road work, accidents, etc.
- The **Transient Dynamic Layer** represents intervals of less than a few minutes, e.g., debris or local weather events.
- The **Highly Dynamic Layer** describes objects that change within seconds or less, such as traffic participants or pedestrians.

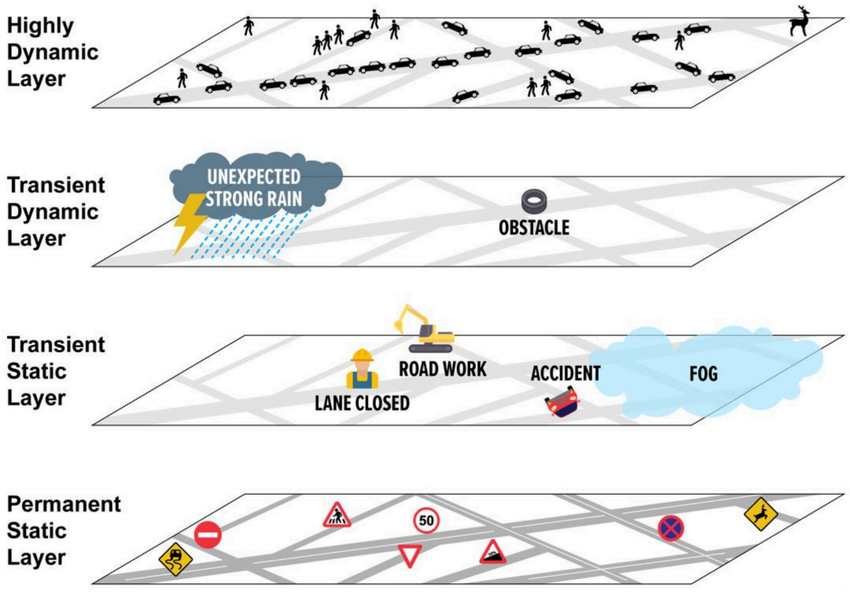


Figure 2.6: HD map layers as defined by the AECC [191].

In contrast to this time-based approach, the commercial mapping company *HERE* defines the layers of an HD map based on the level of detail and purpose [192]. Compared to the model from the AECC, no dynamic layers exist. In this model, the HD map is divided into three layers:

- The **Road Model** is the foundation and is used for navigation. It represents the road network, including its elevation and other characteristics, through a series of ordered polylines.
- The **Lane Model** carries information on lane level, such as lane types, markings, or speed limits. It assists in the decision-making of the vehicle.
- The **Localization Model** includes roadside elements such as signs, barriers, or poles, which can also support localization within the map.

Requirements

Independently of the exact structure of the map, every map has to fulfill specific requirements for efficient use in AD. Poggenhans et al. [193] described the prerequisites for HD maps for AD. The authors derived requirements based on the downstream task that relies on the map data:

- **Routing:** The road network with lane-level precision, including possibilities of lane changes and restrictions for certain traffic participants.
- **Behavior Planning:** Detailed traffic regulations and information on the right-of-way.
- **Behavior Prediction:** Traffic regulations to predict appropriate and regulation-following behavior of surrounding vehicles.
- **Path Planning:** The exact geometry of each lane is described by the left and right boundary. Localization and actuator uncertainties can be taken into account. Also, areas for special maneuvers such as parking or emergency stops.
- **Localization:** Observable elements to enable reliable localization within the map. The map should contain as many features as possible to account for vehicles with different sensor configurations.
- **All of the above:** Ensure that the map is up-to-date. Only by this can the map be used as a reliable input for behavior, trajectory planning, and perception.

To fulfill all of these requirements (completeness, accuracy, and up-to-dateness), a modular and scalable framework is required that allows easy expandability and modifiability of the map [193].

Frameworks

OpenStreetMap (OSM) is a collaborative effort to create a free and editable SD map of the world. It allows users to view, edit, and use geographical data from any location on Earth. A community of mappers who contribute and maintain data on roads, trails, points of interest, railway stations, and more worldwide has created and constantly updated OSM. It offers a free and open-source alternative to proprietary mapping services such as *Google Maps*. OSM provides comprehensive and up-to-date geographic information available to everyone. However, the quality of the map depends on the mapping community [194].

For the representation of HD maps, the two common open-source frameworks are *OpenDRIVE* and *Lanelet2*. These offer the free download of vector road maps.

OpenDRIVE:
In 2005, the Association for Standardization of Automation and Measuring Systems (ASAM) introduced the

OpenDRIVE standard format, which has been continuously updated since then [195]. It uses a data format based on Extensible Markup Language (XML) with a *.xodr* file extension to store the map data. The map format is divided into three layers: The base layer is the reference line that forms the core element of every road. Each reference line is determined by points and is described by a geometric shape such as a line, an arc, or a polynomial, which determines the route and direction of the road. All other objects and features refer to the reference lines. The second layer represents the lanes of the road. Each road has at least one lane, while the maximum number of lanes is unlimited. A center lane with a width of 0 for each road defines the driving direction. The last layer describes features such as signals along the road that correspond to traffic rules and are also associated with a reference line. [186, 195]

Lanelet2:
The second framework for HD maps is the *Lanelet2* format, which also uses an XML data format [193]. The file extension is *.osm*, also used by OSM. Similarly to *OpenDRIVE*, the structure of *Lanelet2* consists of three layers. The physical layer defines real observable objects represented by points and linestrings. The second layer is called the relational layer and connects the elements of the physical layer to lanelets, areas, and traffic rules. Lastly, the topological layer provides additional context information and relationships for the relational layer. In summary, the *Lanelet2* format contains five basic elements, called primitives, which are briefly explained below.

- **Points** are the core element of the map and are defined by their 3D position. They are the only map elements containing positional data.
- **Linestrings** are ordered collections of points with linear interpolation between. These sequences are used to represent the shapes of map elements.
- **Lanelets** are sections of the map for directed motion, such as lanes, pedestrian crossings, or rails. Traffic regulations and topological connections with other elements within a lanelet are not altered. Lanelets are composed of a limiting linestring on each side and have regulatory elements to represent traffic rules.
- **Areas** are sections intended for undirected movements, such as parking lots or green spaces. One or more linestrings define them. They can include regulatory elements.
- **Regulatory elements** describe traffic regulations such as speed limits or right-of-way. Regulatory elements can have dynamic characteristics, indicating that a rule depends on a condition. For instance, speed limits can vary depending on the time of day.

Both map formats, *OpenDRIVE* and *Lanelet2*, contain similar information and thus are convertible to each other. They have individual pros and cons. A detailed comparison of the standards and their application can be found in [196].

2.3.2 Challenges

This chapter outlines the key challenges based on the available HD map formats and the requirements for HD maps.

Map Accuracy:
One key challenge is to build maps with sufficient accuracy. To be usable for all downstream tasks (localization, path planning, motion prediction, etc.), the required precision is in a range of $10\,\mathrm{cm}$ to $20\,\mathrm{cm}$ [189, 197, 198]. This requires highly accurate mapping algorithms for the map used for localization (point cloud, feature, etc.) but also for the semantic map (*OpenDRIVE*, *Lanelet2*). Also, the alignment of these map layers needs to be on cm-level, and they have to be accurately georeferenced for online GNSS-localization. To allow such precision, the spatial resolution of the map also needs to be sufficient [199].

Real-Time Updating and Dynamic Environments:
There are different reasons why real-world traffic environments change from time to time. Some of them can be detectable online by AVs (such as construction sites), and others are hard to detect (inconspicuous changes, mapping errors). Due to these constant changes, the HD maps need to be continuously updated to be used as reliable information sources. This is not feasible if carried out with specially equipped dedicated mapping vehicles. Therefore, algorithms that can leverage the information from all AVs to continuously update HD maps are needed. [200]

Scalability and Coverage:
A related challenge is scalability and map coverage. To completely map large areas and increase coverage, there is a need for more efficient mapping algorithms and the fusion of several data sources. Again, this cannot be done only by using dedicated mapping vehicles. A common idea to tackle this challenge is to use crowd-sourced data, such as OSM, or human-driven vehicle data [201–203]. Other approaches use different data sources, for example, areal imagery [204].

Costs and Effort:
Significant parts of the above-mentioned challenges can be summarized as cost and effort. To implement autonomous driving on a large scale, it is necessary to have commercially viable business models that can be implemented and realized. Thus, the cost and effort to generate, distribute, and constantly update HD maps must be significantly reduced [205]. To minimize costs and increase the scalability of mapping approaches, it is vital to have as few manual steps as possible. This strengthens the need for well-generalizing algorithms that work well in different environments and ODDs [189].

Security, Privacy and Legal Considerations:
Implementing HD maps in autonomous driving raises several security, privacy, and legal issues. Privacy is a significant concern, as these maps can contain confidential information about private properties or individual movements. Legally, the ownership and use of this data can be controversial, raising questions about responsibility for any inaccuracies or misuse. Establishing a legal framework that protects stakeholders while encouraging innovation is difficult, as legislation often lags behind technological advances. It is essential to address these issues to gain public trust and promote the widespread use of autonomous driving technology. [205, 206]

Standardization and Interoperability:
As all organizations working on AD face the issues covered here, standardization can greatly support solving them. The importance of standards cannot be overstated, as they provide individuals and organizations with a shared understanding, making it easier to communicate, collaborate, and adhere to laws and regulations [196]. Currently, all manufacturers use their proprietary mapping formats and processes, and research mainly uses *OpenDRIVE* and *Lanelet2* [205, 207]. There is no common database for HD maps and already mapped areas, leading to duplicate work and insufficient inter-operable data. This also leads to higher expenses for each organization or company and slower development progress. The development of a unified national or international standard could be a catalyst for a shared mapping ecosystem. This could be achieved through collaboration between government agencies, research institutes and universities, private entities, and automobile manufacturers, resulting in a single, reliable, and impartial map data source. This would not only improve the accuracy and compatibility of data sources but also bring economic advantages by making mapping a universally available service. [205, 208]

2.3.3 HD Mapping Approaches

This section reviews existing approaches to solve the challenges in HD map creation. Therefore, approaches to detect roads based on different data are introduced, and based on that, approaches to create, update, and use HD maps are presented.

Road Detection

To create HD maps, the first necessary step is to detect roads in sensor data. This can be done with LiDAR, camera, or sensor fusion approaches. Also, external data, such as aerial images from satellites or planes, can be used to detect roads.

LiDAR-Based Methods:
A method to extract road boundaries from point cloud data is **curb detection**. However, not all roads are bounded by curbs on both sides. There are three approaches to searching 3D point clouds for curbs:

- Based on saliency maps [209, 210]
- Based on voxels: detection of gradients within voxels [211–213]
- Search individual LiDAR scanlines for discontinuities [214–217]

Other approaches use statistical properties of the road surfaces for **statistical road detection**. Therefore, distinctive features with respect to intensity, point density, and environmental characteristics are used to differentiate between road and non-road points [218–221].

Road markings can provide good information for HD mapping when available. Therefore, some approaches for **road marking extraction** from LiDAR data have been published already. Usually, these use the point cloud as a 2D image from the BEV. As road markings are highly reflective, intensity thresholding can be used to extract only these points [222–226]. In the next step, detected regions of high reflectivity are clustered into single road marking elements, such as lines or arrows. Some algorithms apply an additional heuristic filtering step [222, 223]. Deep learning-based approaches have also shown convincing results for detecting road marking in lidar point clouds [227].

Another approach to extracting road points from LiDAR scans is to use 3D semantic segmentation approaches. Famous semantic segmentation algorithms for 3D point clouds are *PointNet* [228], *PointNet++* [229], *RangeNet++* [230], *PointSeg* [231] and the works by Wang et al. [232], and Rochan et al. [233]. However, these approaches depend highly on the training data and hardly generalize to other road types, sensors, etc.

Camera-Based Methods:
Several approaches have also been published for road detection in camera images. A basic and limited approach based on conventional CV uses *Hough*-transformation [234] to detect bright road markings [235, 236].

Most of the recently published approaches use deep learning-based methods for road detection in 2D camera images. This category comprises **row-wise** and **anchor-based** methods. The row-wise methods first partition the image into a grid. A neural network is trained to predict the probability that each grid cell contains at least a portion of a lane [237–239]. Instead of using 2D boxes, anchor-based detection methods use a line defined by two parameters: the origin point at the lower edge of the image and an angle θ that dictates the slope of the line. The anchor can be placed vertically ($\theta = 90°$) or in a perspective way, with the latter showing better results in the available publications. Algorithms that implement the anchor-based approach are *PointLaneNet* [240], *3D-LaneNet* [241], *CurveLane-NAS* [242], and [243].

For road detection in 2D camera images, semantic segmentation algorithms trained with AD datasets can also be used. Currently, the best-performing approaches are based on ViT [244], such as *SegFormer* [245]. For further information and details, the reader is referred to existing survey papers on semantic segmentation in general [246] and especially with ViTs [247].

Sensor Fusion-Based Methods:
To overcome the limitations of the described approaches, sensor fusion can help. These approaches have higher requirements for the data and need synchronized and precisely calibrated LiDAR and camera data. To take into account the entire context of the environment, some methods implemented Conditional Random Fields (CRFs) to fuse LiDAR and camera road detection results [248, 249]. Other approaches are based on Fully Connected Network (FCN) for information fusion of LiDAR and camera [250, 251].

Aerial Imagery-Based Methods:
In addition to onboard sensor data, road markings or direct road surfaces can also be detected in aerial imagery. The best working approaches use supervised neural networks trained to detect road markings or roads. Different algorithms are based on feature pyramids [252, 253], self-attentive networks [254], or Convolutional Neural Networks (CNNs) [255, 256]. Yang et al. [257] presented a non-deep learning approach. It is semi-automated and combines manually chosen road points with a region-growing algorithm in combination with morphological and vector operations. The approach promises to generalize better to different applications, as it is less data-sensitive. But, human interaction is needed for this algorithm.

The presented approaches in the field of aerial imagery-based road detection have shown promising results. However, there are some limitations. The algorithms evaluated are based on proprietary datasets for training the extraction of lane markings or roads. To make these more scalable and applicable, it is preferable to use open-source map data instead of limiting them to small, nonpublic datasets. Deep learning algorithms especially tend to show strong domain specificity, demonstrating performance dependent on the context in which they were trained, such as specific geographical regions like the United States or Germany.

HD-Map Creation

The road information extracted with the presented algorithms must be further processed to receive not only roads but comprehensive HD maps. The following section will differentiate between “Offline Static Mapping” and “Online Local Mapping”. The first presents algorithms to create semantic HD maps as in Subsection 2.3.1, and the latter is a novel approach without offline generated HD maps.

Offline Static Mapping:
Gwon et al. [258] presented a pipeline to generate semantic lane-level maps from raw LiDAR scans and GNSS data. Their approach extracts road markings based on the intensity values of accumulated point clouds. A new road representation is introduced, modeling roads as sets of piecewise polynomial curves. A similar LiDAR-based approach was published by Yu et al. [259].

Other approaches use aerial images to extract accurate vectorized maps. [260] introduced a mapping pipeline based on aerial photos. First, lane boundaries in non-section areas are detected, and later, they are heuristically connected. An overview of how Unmanned Aerial Vehicles (UAVs) can support HD mapping and make it more scalable was published in [204].

Another common approach is to use crowd-sourced vehicle data, mainly GNSS. Although each individual data point is noisy and limited in accuracy and information content, with enough data, precise HD mapping results can be obtained [202]. The company *Lyft* uses OSM data for initial estimation of HD maps and improves it with real-time vehicle data [6]. This allows for continuous map updates and considering changes, such as construction sites.

Only one publication is available that includes the entire mapping pipeline [188]. The authors presented the whole approach, from raw sensor data to HD maps, including point cloud and vector maps. The authors stated that many manual steps were required to obtain satisfactory results, especially for the vector map. The code used is not published, and thus, the performance of this approach cannot be tested on other data sources and environments. This underlines the need for research in the field of applicable and scalable HD mapping pipelines.

Online Local Mapping:
In recent years, new techniques for the online creation of local HD maps have been developed that combine LiDAR and camera-based approaches presented in Subsubsection 2.3.3. Known as end-to-end methods, these strategies are based solely on neural networks to extract features and create maps depicting the ego vehicle's surroundings. However, in contrast to the offline methods described above, these methods do not generate static maps but live maps of the direct environment, supposing the ego as the center of the local map. This means that not only static but also dynamic objects, such as other vehicles, can be directly added to the map. Most of these approaches use BEV to represent the data and the output maps. Despite the increasing popularity of BEV, there are challenges in maintaining the unique strengths of each modality during data transformation.

Several approaches, including *HDMapNet* [261], *VectorMapNet* [262], *MapTR* [263] and *MapTRv2* [264], have been developed to automatically and online generate HD maps. These methods fuse camera and LiDAR data to map environmental features such as lanes and road boundaries. This procedure includes encoding the features of images and point clouds and translating them into a unified BEV representation. Finally, the features are decoded into vectorized maps. *HDMapNet* requires an extra step to merge the outputs, while *VectorMapNet* and its refinements, *MapTR*, and *MapTRv2*, directly create polylines. *MapTR* and *MapTRv2* enhanced scalability and modeling of complex map features. *SuperFusion* [265] recently achieved convincing results on the *nuScenes* dataset [45]. The authors focused on an increased range for stable local mapping, as this is the requirement of the subsequent path-planning algorithm. However, the approach only works when both LiDAR and camera data are available. *InstaGraM* [266] is a method based solely on camera images, not using LiDAR. The polylines of the map are modeled as graphs. This allows efficient real-time computation without heuristic post-processing. Ding et al. [267] recently published *PivotNet* that outperformed all other approaches only using camera images. It is based on transformers for BEV feature encoding and a line-aware point decoder. Opposing the trend of focusing on vision-based approaches, *LiDAR2Map* [268] uses camera images only during the training process to distill camera information. Online, during inference, only data from LiDAR are used for map prediction. This method could outperform other LiDAR- and even camera-based approaches. In [269] and [270], the authors combine the semantic segmentation of LiDAR and camera to obtain a semantic map of the local environment. *BEVFusion* [271] (not to be confused with the other *BEVFusion* [272]) goes one step further and combines local semantic mapping with 3D object detection. Another similar approach to creating semantic maps that include dynamic objects is *Bi-Mapper* [273].

Other approaches expanded the idea of a more integrated perception system. As lane detection or online HD mapping and object detection are not uncorrelated tasks, they can benefit from each other. This leads to the novel task of BEV semantic segmentation, segmenting the local environment in roads, sidewalks, vehicles, etc. Path-planning algorithms can directly use this information to plan appropriate behavior. *Simple-BEV* [274] fuses cameras, RaDARs, and LiDARs for semantic BEV map prediction. This approach can produce accurate results even with a simple 2D-to-BEV projection. *SkyEye* [275] is a self-supervised approach to predict local semantic maps around the ego vehicle, including static and dynamic parts. To overcome limitations due to the high data requirements of the previously presented approaches, two measures are taken: First, the approach is self-supervised and only needs 2D annotated images and no BEV ground truth. Second, only the front view camera is used to reduce the data needed for further training.

A rapidly evolving field is 3D semantic prediction to not lose the three-dimensional context of the data. Therefore, most approaches voxelize the surrounding and predict a semantic class for each voxel [276–281].

The following sections will focus on static HD maps for autonomous vehicles and show their application and usage.

HD-Map Updates

As mentioned in Subsection 2.3.2, a significant challenge is to keep created HD maps up-to-date. Otherwise, they cannot provide reliable information for AVs.

Many published approaches in research and also from the commercial provider *Lyft* [6] use crowd-sourced lightweight vehicle information such as GNSS tracks. Pannen et al. [200, 282] introduced a particle filter-based approach to detect changes in the map. This happens during particle-filter-based localization. By the distribution of particles and probabilities, an estimate of how well the map represents the actual environment can be made. With many available position and trajectory data, even with high noise, changes in road topology can be detected [203]. *Lyft* uses crowd-sourced vehicle data for continuous map refinement [6]. Different methods of how to use these data for map updates have been investigated. For example, using blockchain technology for crowd-sourced map updates [283]. To avoid data protection issues when using vehicle position data, federated HD map updating can be implemented for protected privacy [206].

Kim et al. [284] introduced a method to update the point cloud layer of HD maps. Therefore, when a change is detected in the LiDAR localization pipeline, the part is newly mapped using LiDAR SLAM, and the point cloud map part is sent to the cloud to merge with the whole map. A similar approach is *Simultaneous Localization And Map Change Update (SLAMCU)* [285]. Berrio et al. [286] developed and validated a long-term map maintenance pipeline. They introduce new layers to the map for newly mapped and added features. These layers are then overlayed with the original map. The authors also corrected the original maps based on detections and viewpoints of the mapped features.

For further details on different map update approaches, survey papers are available [287, 288].

HD-Map Usage

There are two prominent online use cases for HD maps: As support for different perception tasks, such as localization or object detection, and as decision criteria for behavior planning. These will be elaborated on in the following. Figure 2.7 depicts all components of an AD software stack (as presented in Section 1.2) and how the HD map is used throughout the stack.

Support for Perception:
The most common use case is to use the point cloud layer of the HD map for LiDAR localization. Therefore, the current scans of the onboard LiDAR sensor are registered against the offline built map, for example, using an ICP-based approach [289] to determine the ego position. *LOL* [290] adds some refinements for a more robust and precise localization with pre-built maps. Other approaches use detected features or objects to estimate the current transformation between the ego and the map frame [291].

Also, the semantic vector map can be used for localization. When done robustly and reliably, the heavy and memory-intensive localization layer can be omitted [292]. Therefore, semantic features are detected in the current sensor frame (usually a camera or LiDAR) and matched with semantic features in the HD map. In [292], an approach for localization with LiDAR was presented on semantic maps without a point cloud layer. Specific features can be detected in 2D camera images and compared to the features of the HD map to find correspondences [293]. [294] detects lane markings in LiDAR point clouds and traffic signs in the

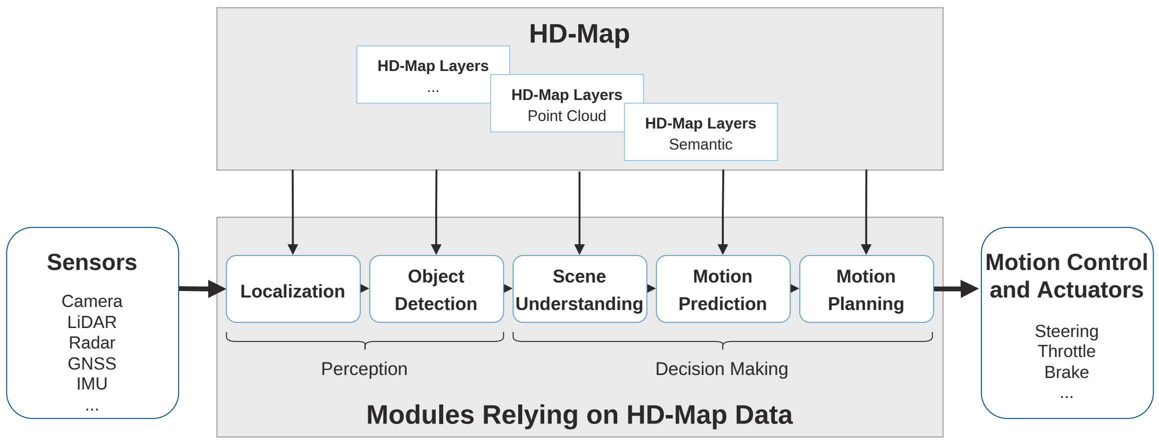


Figure 2.7: Usage of HD maps across the AD software stack. Adapted from [287].

corresponding camera images. These detections are compared with a semantic map and fused with GNSS and IMU in a probabilistic algorithm for precise localization.

Neural Map Prior (NMP) [295] combines online local mapping with updating offline created HD maps. The main idea is to progressively combine global map prior updates and local map inference for each frame. Therefore, an attention-based deep learning approach was chosen, and the map was used in a sparse neural map tile representation for efficiency reasons.

HD maps can also be used for enhanced object detection. On the one hand side, detecting static objects, such as traffic signs, road markings, etc., can be improved. This also improves the estimation of the drivable space without any (static or dynamic) objects [296]. On the other hand, direct detection of dynamic objects, such as other vehicles, can also benefit from prior information from HD maps, such as the road layout [297–299].

Criteria for Decision Making:
Finally, HD maps are also widely used for non-perception tasks in autonomous driving, as depicted in Figure 2.7. Scene understanding is the interface between perception and the rest of the software stack. The objective is to comprehensively understand the environment and the relations between everything detected in the perception modules [300]. Therefore, HD maps can be of great use, as they provide static information that can be used to improve the understanding of the surroundings, such as pedestrian crossings, traffic signs, etc. [301].

As the environment of AVs is highly dynamic, it is not enough to only detect the surrounding objects, but they also have to be predicted. Motion prediction refers to the task of estimating the future behavior of road participants given their current states and a model of their static environment [287]. Thus, it is beneficial and improves the performance of motion prediction algorithms to use an HD map as an additional input [302–304].

For the usual path planning approaches, HD maps are used to know the static driving space and additional information relevant to the task, for example, speed limits and regulations. The map's geometry is used to define constraints which, combined with the dynamic objects detected and predicted, create the problem to be solved by the path planner [305, 306].

In summary, one can see the importance of HD maps for the current state of the art in AD. Therefore, much research has been conducted recently on generating, updating, and using them efficiently. However, many remaining open challenges will be addressed in the following part of this thesis.

3 Problem Statement

Extending the findings of the related work (Chapter 2), this chapter draws conclusions that lead to open challenges. Based on this, the main Research Question (RQ) of this thesis is defined, and the corresponding Sub-Questions (SQs) are derived. These are addressed in the remainder of this work.

3.1 Conclusions from Related Work

First, the state of the art is briefly concluded, and some critical remarks are drawn. In the next section, RQs are derived.

Conclusion 1: SLAM and map-based localization are important parts of AD software stacks.

SLAM is a key component of robotic systems in general and AD in particular. It is needed to generate maps that can be used for map-based localization. Most modern software stacks for AD are based on map data usage. Thus, research in this field has been conducted for many years. However, all approaches still have limitations in different fields, stemming from input sensor data, hardware limitations, environmental influences (such as weather), etc. Thus, SLAM applications are usually built for specific applications, and their generalization is insufficient. In the context of this work, two specific applications for the following vehicle platforms had to be solved:

- **AV-21**, the autonomous racecar for the IAC (Subsection 4.1.1)
- **EDGAR**, TUM's autonomous level 5 research vehicle (Section 5.1)

Critical remark: No existing approaches were suitable for our specific applications.

None of the methods from related work could be applied for the two applications [11, 12, 18]. This is because the vehicle platforms and the environments differ significantly from the data collection platforms of standard datasets for AD. Usually, published algorithms are validated and tested on these. The sensor setups of our vehicles include multiple LiDARs, in the case of the AV-21, without overlapping FOV, and lacking highly precise time synchronization. Additionally, the AV-21 drives at low-feature race track environments and high speeds. Section 2.2 showed that sensor fusion can efficiently overcome the limitations of monomodal SLAM approaches. There is a need for localization algorithms that work with different multimodal sensor setups and can handle imperfections in the data.

In autonomous racing, many challenges stem from the high velocities. These lead to larger baselines between the individual frames, making it hard to precisely and robustly estimate the transformation between consecutive data frames. Moreover, high velocities aggravate the negative effects of erroneous sensor calibration and time synchronization. Especially on oval race tracks and long straights, algorithms have to deal with sparse features that can be detected and matched. The environments tend to be repetitive which can lead to a loss of tracking. Another challenge comes from the limited computational resources and the real-time requirements. As the entire autonomous racing software stack runs on one single computer in the vehicle, the localization approach needs to be resource-saving but still be able to use all sensor information in real-time. Currently, there are no algorithms that can solve all of these challenges and can reliably solve the localization for our applications.

Conclusion 2: AD software stacks are highly dependent on HD maps.

As Section 2.3 revealed, most of the software modules in AD rely on HD maps. Therefore, most research papers assume error-free and up-to-date maps. The creation of these maps is no big part of the current research, which focuses on local semantic mapping to detect the direct environment of the ego. There is still no entire HD mapping pipeline available. Research papers using HD maps focus on the downstream application, not the map creation process. In the current state, only commercial providers such as *Nvidia* [8], *Mobileye* [7], or *Lyft* [6] have the tools to create HD maps with small manual processing.

Critical remark: There is a lack of pipelines to create and handle HD maps in the research community.

In the real world, traffic environments can change quickly; mapping errors can occur, and HD maps may not cover all the roads in the mapped area. It is not feasible to map entire cities with dedicated mapping systems. Therefore, the creation and updating of HD maps must be based on highly available data, e.g., from normal vehicles on the streets or, in the future, from automated vehicles. Although there has been research in this field, there is still no applicable and generalizing pipeline for the entire life cycle of HD maps. The main challenges are scalability, robustness, topicality of maps, generalization, and general applicability. Research should focus not only on long-term solutions for AD but also on HD maps that represent the current state of the art. Novel mapping pipelines are needed to quickly generate new HD maps. These maps can then be subsequently replaced or supplemented by online mapping approaches.

In the concrete application of the EDGAR research vehicle, a mapping pipeline is needed that allows quick mapping of new areas with limited human post-processing. It needs to use the sensor data from the research vehicle and publicly available data, such as OSM. Only if these requirements are fulfilled, the whole research team can profit from the outcomes. This mapping pipeline and the ability to quickly create high-quality maps of different regions allow researchers to extend their specific approaches to new applications, not only focusing on standard perception datasets with included HD maps. There is a need for open-source mapping tools to close the gap between the aforementioned commercial providers and the research community. Currently, no open-source mapping pipeline exists that allows to create maps with any research vehicle and use them for research purposes. This thesis tries to reduce this gap by providing a mapping pipeline that is capable of creating maps that follow open standards to be easily used in several research applications.

3.2 Research Questions and Methodology

Based on the conclusions drawn and the critical remarks derived, RQs are defined. The main RQ to be answered brings the critical remarks and open challenges from related work together. The superordinated goal of the following chapters is to answer it as comprehensively as possible.

RQ: How to formulate an open generic framework for robust and applicable HD mapping and localization for AVs?

The RQ is derived from a critical analysis of the current state of the art. It revealed a gap in the development of a universally applicable and efficient system for localization in complex environments and applied HD mapping. The current landscape of AV research shows significant advances in both localization and mapping. However, these systems often face applicability, robustness, and effectiveness challenges, especially in diverse and dynamically changing environments. Some solutions excel in well-structured urban areas, but their performance may degrade in more challenging settings, such as rural or unstructured environments.

Approaches are specifically tailored for a single application and there are no adaptable frameworks available. This thesis aims to design and evaluate a generic pipeline that addresses these shortcomings. This involves integrating advanced sensor fusion and data processing techniques to achieve high accuracy and robustness in localization and to create up-to-date and detailed HD maps. The term generic implies that the approach needs to be designed in a way to work with different vehicle and sensor platforms. To work across different ODDs, such as racing and urban driving, it needs to be robust against different environmental conditions. As the approach will be part of different research platforms, it has to be easily applicable so that it can be transferred to novel applications and extended with new approaches. To answer this RQ, it is broken down into two subquestions.

SQ1: How can sensor fusion be used for localization and mapping algorithms in difficult real-world applications?

First, the focus is put on the location of the race car AV-21 built for the IAC. Since existing methods could not solve this task, new approaches are introduced in Section 4.1. Sensor fusion is found to be an efficient technique for overcoming sensor limitations and handling this challenging application. The developed algorithms are then evolved, tested, and validated for public traffic scenarios to manage the transfer of approaches from closed race tracks to public roads.

In Section 5.3, a localization algorithm is presented for the multi-LiDAR setup of the research vehicle EDGAR. It solves the previously mentioned challenges of the sensor setup, faulty calibration, and missing precise time synchronization. This algorithm is also the basis for the HD mapping pipeline.

SQ2: How to design an open HD mapping pipeline for AVs?

The just mentioned multi-LiDAR-based localization approach for EDGAR builds the foundation for a point cloud mapping algorithm. This allows the building of precise point cloud maps that can be used as a localization layer of an HD map. On top of this, different methodologies for vector map building are presented. Vector maps include semantic information, such as lane topologies, traffic rules, etc., and are used for motion planning and prediction. The algorithms aim to be as automated as possible to meet the goal of an applicable mapping approach with limited manual processing. For a quick map creation and an easy update process, crowd-sourced data (mainly OSM) are evaluated and integrated. Finally, a tool to merge the data and create a consistent HD map is presented to allow applicability within the software stack. This thesis presents an entire HD mapping pipeline that can be applied to different vehicle platforms. To build a solid base for future research across different applications and projects, the pipeline is designed openly, meaning the code, used algorithms, and data are open source.

To bring the two subquestions together, in the end, an approach is presented to integrate localization modules for autonomous racecars and the HD mapping pipeline into a common framework.

4 Sensor Fusion for Localization and Mapping in Diverse Environments

Parts of this chapter have been published in [11, 12, 14, 16, 307].

This chapter addresses the first research subquestion:

SQ1: How can sensor fusion be used for localization and mapping algorithms in difficult real-world applications?

Therefore, first, the localization approach developed for the IAC is presented. Building upon the results, the following approaches will gradually try to transfer the findings and methods toward public roads.

4.1 High-Speed Localization

Localization of racecars at high speeds on race tracks is a crucial but challenging task in autonomous racing. The main challenges arise from two points: high velocities and the environment. High speeds lead to increased noise in sensor data, such as distortions through motion blur in LiDARs and cameras. Since sensor data are recorded at fixed frame rates, the distance driven between the frames is proportional to the velocity. Compared to urban regions, for example, the environment on race tracks is usually relatively monotonous and repetitive. This particularly applies to oval race tracks, such as the Indianapolis Motor Speedway (IMS), where the initial race of the IAC took place. Moreover, the requirements for the localization algorithm are high as unprecise or unstable ego pose estimates can lead to a total loss of the vehicle.

4.1.1 Vehicle - AV-21

This chapter aims to give the necessary background to understand the requirements and general conditions. First, the IAC is briefly described, and then the vehicle platform, the AV-21, and its sensor setup and compute platform are presented.

Indy Autonomous Challenge

Each participating team in the IAC received an identical copy of the specially developed autonomous race car, the *Dallara* AV-21. Thus, the hardware and sensor setup was predefined and can be seen as a framework condition for the algorithmic development. Due to the tight schedule of the challenge, algorithms had to be developed before the exact hardware specification was frozen and real-world data were available. Thus, the simulation of vehicles, sensor data, and the environment was critical in successfully participating in the final competition. The IAC took place on October 23, 2021, in Indianapolis, and the follow-up event, the Autonomous Challenge at Consumer Electronics Show Las Vegas (AC@CES), was held during the Consumer Electronics Show (CES) in Las Vegas on January 07, 2022. For more details on these events, rules, the whole TUM software stack, and the results, the reader is referred to the related publications to which this thesis contributed [13, 307]. A more detailed consideration of the perception system, limitations, and lessons learned have been published in [12].

The perception data from different teams was standardized and published as the *RACECAR* dataset [55]. The work in the context of this thesis contributed to this publication.

Vehicle Platform: AV-21

The *Dallara* AV-21 uses a *Dallara* IL-15 chassis, which is known from the Indy Lights Series [308]. Figure 4.1 shows a photo of TUM's AV-21 avoiding static objects on the IMS. The driver seat, steering wheel, and pedals were removed to create space for the computer, sensors, and other electronic components (battery, low-level controller, etc.).

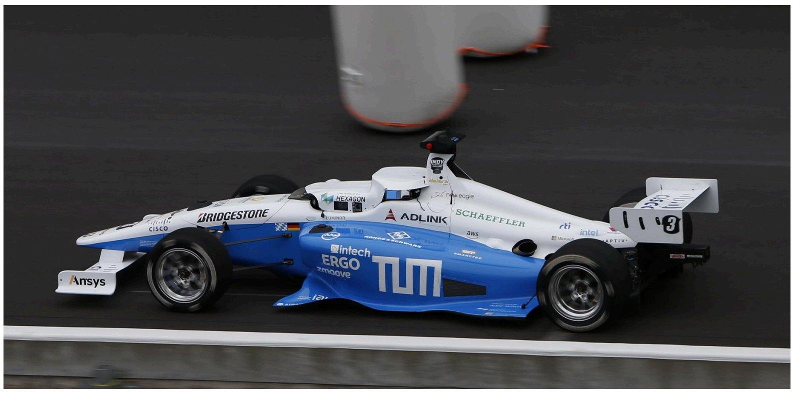


Figure 4.1: TUM's AV-21 during the single-vehicle competition on October 23, 2021.

Table 4.1 gives an overview of the components installed in the autonomous system of the AV-21.

Table 4.1: The perception components of the *Dallara* AV-21. The FOV is given horizontally and vertically (h x v) [12].

Device	Manufacturer	Model	Frame rate	FOV (h x v)
2x GNSS Receiver	*NovAtel*	*PwrPak7D* Receiver	20 Hz GNSS, 100 Hz IMU	-
3x LiDAR	*Luminar*	*H3*	1 Hz - 30 Hz	120° x 0° - 30°
2x Front Camera	*Allied Vision*	*Mako G319C*, 12 mm FL	up to 37.6 Hz at full res.	34° x 24°
4x Side Camera	*Allied Vision*	*Mako G319C*, 3.5 mm FL	up to 37.6 Hz at full res.	102.8° x 77°
2x Side RaDAR	*Aptiv*	*MRR*	10 Hz	90° x 5°
1x Front RaDAR	*Aptiv*	*ESR 2.5*	10 Hz	90° x 4.4° (short range), 20° x 4.4° (long range)
Computing Platform	*ADLink*	*AVA-3501*	-	-
Network Switch	*Cisco*	*IE500*	-	-

The AV-21 is equipped with various sensors to cover different software approaches and provide sufficient data. Two *NovAtel PwrPak7D* GNSS receivers with RTK-correction, dual antenna setup, and an IMU each are installed. Six cameras around the vehicle provide a total 360° view of the surroundings. The cameras that are directed to the sides and rear of the vehicle have a 102.8° horizontal and 77° vertical FOV. An *Edmund Optics* lens with a focal length of 3.5 mm is attached to them. For higher detection ranges, the front cameras are equipped with a longer focal length (12 mm) resulting in a FOV of 34° horizontally and 24° vertically. The two cameras are placed parallel as a stereo setup with a baseline of 24 cm. However, it should be mentioned that the cameras are not calibrated, and the mounting is not rigid enough for conventional stereo usage. Each of the three LiDARs has a horizontal FOV of 120° resulting in 360° total coverage. The vertical coverage can be dynamically configured between $0°$ and $30°$. The AV-21 is also equipped with a total of three RaDARs. The one directed to the front alternates between short- and long-range; those directed to each side are mid-range RaDARs. A graphical overview of the entire sensor setup in BEV can be seen in Figure 4.2. [12]

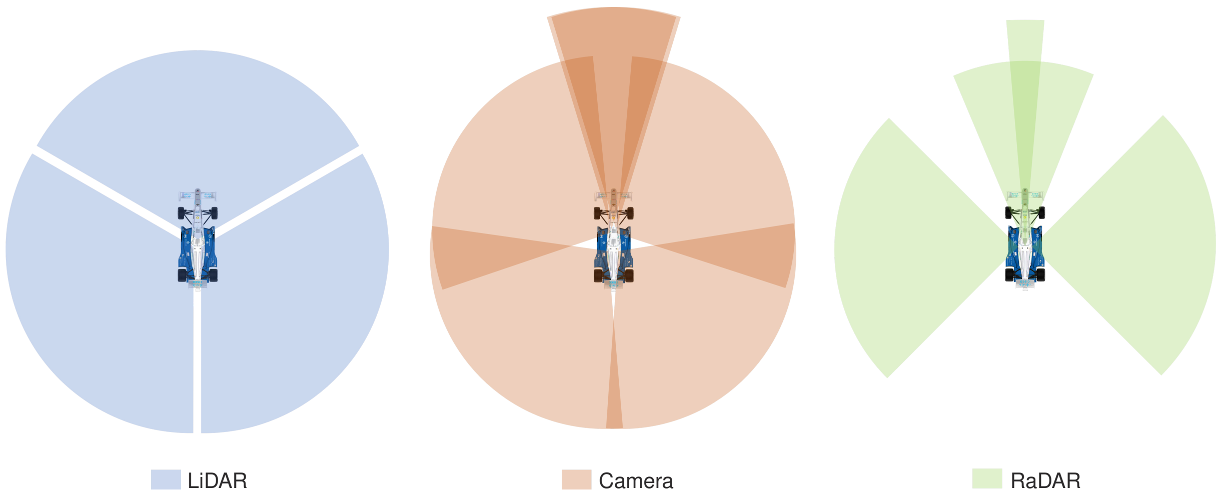


Figure 4.2: Sensor setup and FOV of the IAC racecar AV-21.

4.1.2 LiDAR-based Localization

As all of the approaches previously published for the localization of autonomous race cars used LiDAR sensors, this was the first approach chosen for the IAC [309–311]. However, it has turned out that existing approaches did not work for simulated and later real-world data from oval race tracks, such as the IMS. The following will introduce the LiDAR pipeline and investigate the reasons for the failure of the LiDAR localization approaches.

LiDAR Fusion and Prefiltering

First, the LiDAR pipeline from the single raw point clouds to the published topics in *Robot Operating System 2 (ROS 2)*[1] will be presented. The steps carried out by the LiDAR driver and the preprocessing pipeline are visualized in Figure 4.3. After receiving raw data from one of the three sensors, the driver waits a definable amount of time (e.g., $100\,\mathrm{ms}$ at $10\,\mathrm{Hz}$ sensor frequency) and, if applicable, collects point clouds from the other sensors. After this time, all received points are transformed into a common frame, fused into one point cloud, and published. Subsequent algorithms can use either this full point cloud or the processed point cloud. The preprocessing consists of three steps:

1. Geometric filtering that deletes points in certain parameterizable regions.
2. Voxel filtering to reduce the number of points based on voxelization.
3. Ground filtering to further reduce the point cloud size by removing street surface points.

In particular, the fusion methodology introduces several problems. As in the racecar, no time synchronization was running; the timestamps from the sensor clocks could not be used because their offset was unknown. Therefore, the *ROS 2* timestamps from the computer clock had to be used. These denote the time at which the computer receives the raw data from the sensor via the network switch and Ethernet. That leads to the time

[1] https://ros.org/

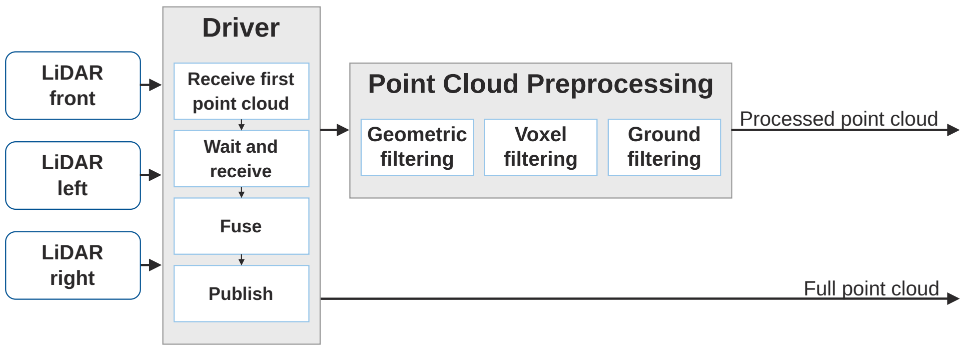


Figure 4.3: Fusion and preprocessing pipeline of the three LiDARs.

interval from the actual data capture to the transmission and finally to the driver being neglected. Additionally, the driver waits up to $100\,\mathrm{ms}$ for the sensors. This has two disadvantages. First, it leads to out-of-date data, which has to be compensated for. At $300\,\mathrm{km\,h^{-1}}$ ($83.3\,\mathrm{m\,s^{-1}}$), the car travels up to $8.33\,\mathrm{m}$ during $100\,\mathrm{ms}$. Second, it leads to a significant time difference between the points of the different sensors, which are published as one single point cloud. Therefore, advanced compensation algorithms are needed, as different time compensations are required for different sections of the fused point cloud. These inconsistencies within the point clouds lead to difficulties in scan matching [55] which finally prevented a robust LiDAR localization on the IMS.

Point Cloud Registration

The performance of different algorithms was investigated in detail to further investigate the problems of point cloud registration. Therefore, the most famous algorithms from the *pcl* [312], *omp* [313], and *fast* [314] libraries were evaluated: ICP (*pcl*), ICP–nl (*pcl*), ICP–normals (*pcl*), GICP (*pcl*), GICP (*fast*), VGICP (*fast*), NDT (*pcl*), NDT (*omp*). The evaluation and comparison pipeline process is presented in Algorithm 1.

Algorithm 1 Scan Registration Parameter Optimization

```
Require: Set Registration Method
 1: for point cloud in samples do
 2:     if cloud == target then
 3:         Sample target prefiltering parameters
 4:         RUN distance filter
 5:         RUN voxel filter
 6:         RUN outlier radius removal
 7:         return Processed target point cloud
 8:     else if cloud == source then
 9:         Sample source prefiltering parameters
10:         RUN distance filter
11:         RUN voxel filter
12:         RUN outlier radius removal
13:         return Processed source point cloud
14:     end if
15:     Registration method query
16:     Set registration parameters according to method
17:     RUN point cloud alignment: processed target and processed source
18:     Save sampled values, final transformation, and runtime
19: end for
```

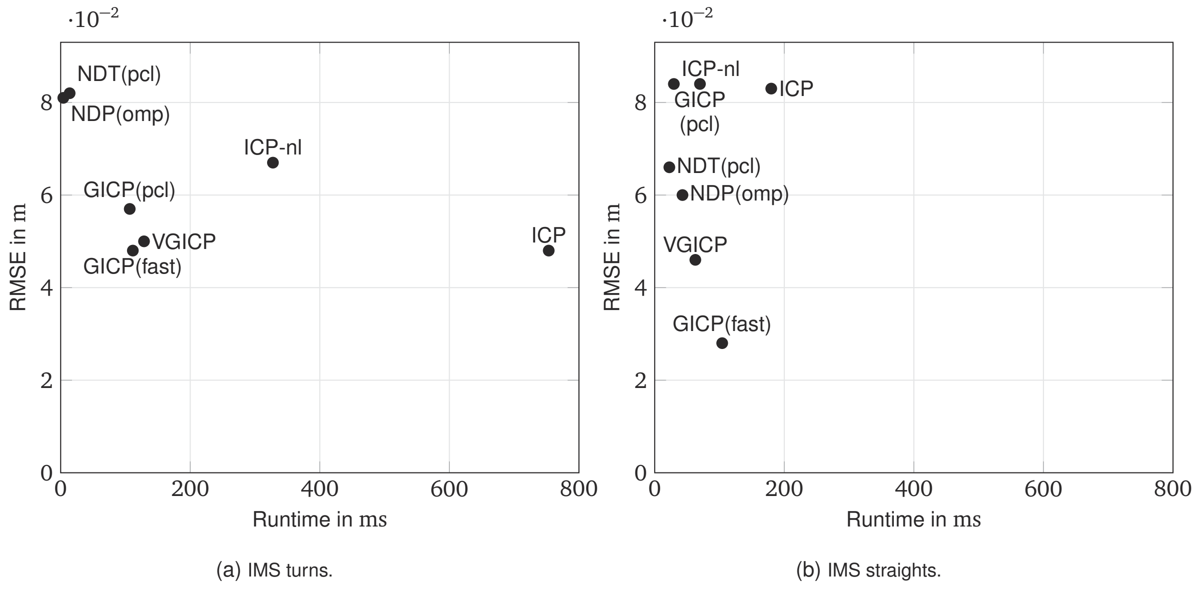


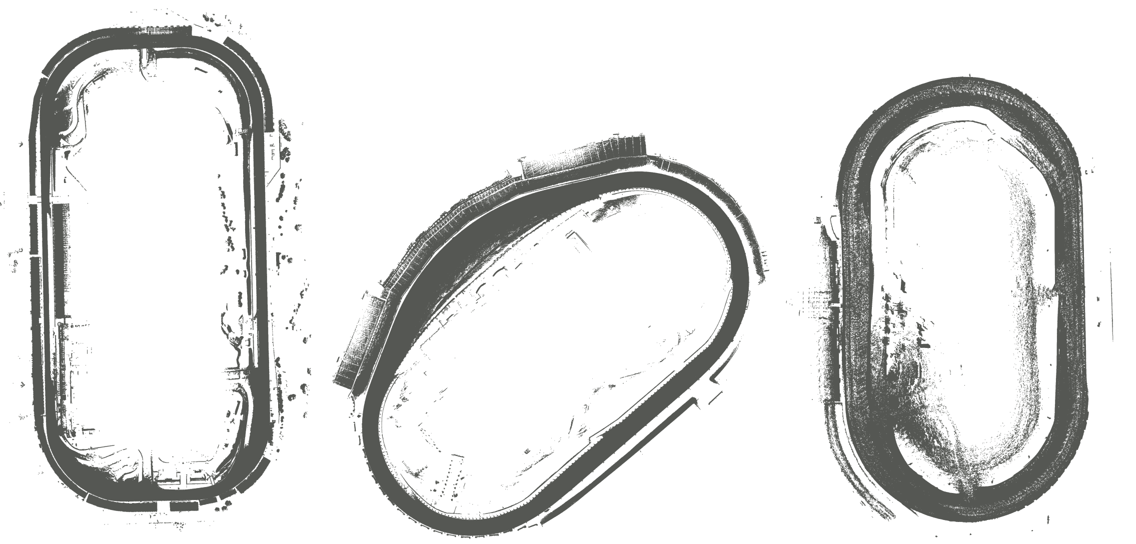
Figure 4.4: Comparison of scan registration methods on the IMS.

The full results of this analysis can be found in [315]. Key findings are briefly summarized below. Because motion compensation did not yield meaningful improvements for the data used and to keep the comparison focused on the registration algorithms, no motion compensation was applied to the point clouds. All experiments have been performed on an *Intel Core i7-9750h* processor. For comparability, all registrations were calculated in a single thread. The code using the above-mentioned libraries was written in *C++*. Figure 4.4 shows the performance of the mentioned scan matching algorithms on the IMS after parameter optimization. The different behavior in turns and straights shows the high dependence on the environmental appearance. Overall, the two GICP methods show the most promising performance in terms of accuracy and runtime. Additionally, they are more robust to suboptimal parameterization, which has to be assumed for realistic applications. All of these parameters were found to be highly dependent on their chosen values. The optimal parameterization depends on the appearance of the frames and thus cannot be determined for all different ODDs. The initial guess shows the strongest sensitivity. A bad initial guess leads to failed pose convergence or exploding runtimes. Therefore, an accurate initial pose estimation is crucial. The NDT algorithms show advantages in terms of runtime and are thus recommended for real-time applications, such as online localization. Another finding is that robustness is more important than accuracy. When using a pre-built map, there will be no global drift, and thus, the most important thing is that there is no loss of tracking in the localization module. [315]

Point Cloud Mapping

These findings led to the successful offline creation of 3D point cloud maps of the oval race tracks driven: IMS, Las Vegas Motor Speedway (LVMS), and Lucas Oil Raceway (LOR). Figure 4.5 shows these point cloud maps in BEV.

The maps were built using the *GICP* implementation from the *fast* library [314] integrated into the *hdl_graph_slam* [316]. This method provides a good trade-off in terms of accuracy and parametrization. The generated maps were manually post-processed with the *interactive_slam* [317] for improved results and removal of errors. This tool allows one to manually add constraints, such as loop closures, to refine graph optimization and, thus, the output map. The GNSS signal with cm-level accuracy was used as the initial guess for each scan registration. Therefore, a precise initial guess along the whole track was guaranteed, and the point cloud scan matching only had to do the fine registration. Without highly precise GNSS data



(a) Indianapolis Motor Speedway (IMS) (b) Las Vegas Motor Speedway (LVMS) (c) Lucas Oil Raceway (LOR) Indianapolis

Figure 4.5: BEV of generated point cloud maps of IAC race tracks. The tracks are individually scaled and do not represent the real size ratios.

and solely based on scan matching, all mapping approaches failed to create globally consistent closed-loop maps.

Map-based Localization

Due to the described challenges, a LiDAR-only localization using the created 3D point cloud maps was impossible. Available open-source algorithms for map-based localization failed directly [55]. To investigate the benefits of LiDAR registration for state estimation, a custom 3D EKF estimator was developed. Due to the high accuracy of the GNSS, the addition of LiDAR scan matching inputs leads to a decreased accuracy. However, when the GNSS signal is degraded, and its standard deviation increases to a few meters, or the signal is completely cut, the estimated state benefits from scan registration. In summary, this approach can increase the robustness of state estimation against short-term GNSS signal losses or outliers. However, other approaches are needed to enable robust and precise localization on oval race tracks at high velocity and without accurate GNSS reception. The following section will present a sensor-fusion-based approach to this problem.

Many problems arise from the multi-LiDAR setup. Missing time synchronization, erroneous calibration, and distortions at high speeds lead to degradation of scan matching. One way to tackle these challenges is to adapt the vehicle configuration. Section 5.3 presents a novel approach for such setups. It was developed for a passenger car, and its adaptability to racecars is currently under investigation. Also, road course race tracks provide better features for localization and mapping. Figure 4.6 shows a 3D map of the *Autodromo Nazionale* in Monza created with the new mapping pipeline. The developed mapping and localization approaches will be investigated further on this track.

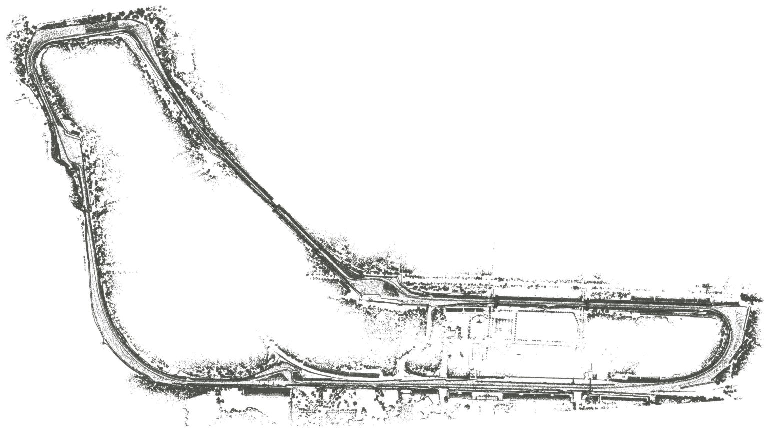


Figure 4.6: BEV of the created 3D point cloud map of the *Autodromo Nazionale* in Monza.

4.1.3 Coordinate Separation for LiDAR-Camera Localization

This chapter is based on [11], published within the scope of this thesis. More details on the implementation, experiments, and results can be found in the publication.

To avoid some of the problems of the previous chapter and enable localization on oval race tracks, a novel approach based on sensor fusion was developed. It aims to combine the advantages of LiDAR and camera for high-speed localization on oval race tracks. Since for the IAC, data could be recorded while teleoperation was performed, it was possible to do mapping runs before going fully autonomous. Therefore, the algorithm presented solves the task of online localization on race tracks at high speeds using offline-built maps. The offline mapping process consists of two steps:

1. Detection of track boundaries and projection on a 2D plane.
2. Visual SLAM to build a feature map via *OpenVSLAM* [125].

The generated data are then used online with the methodology described in the following. The car is assumed to be equipped with precise GNSS for mapping, at least one front-facing monocular camera, and a $360°$ LiDAR. LiDAR SLAM and localization algorithms struggled mainly with monotonous and repeating environments such as on long straights. Camera approaches had problems with increasing velocities and, thus, increasing baselines, leading to decreased accuracy. This approach attempts to avoid these weaknesses by combining the sensor-specific advantages of the camera and LiDAR.

Concept Overview

As the strength of LiDAR proved to be in detecting objects within a range of $\sim 80\,\mathrm{m}$ in 3D, and the camera could also use distant features, for example, on long straights, the presented algorithm aims to combine these advantages. Therefore, the LiDAR task is to detect the wall that limits the outside of the track (Steel and Foam Energy Reduction Barrier (SAFER) barrier) to estimate the lateral position. The camera runs a feature-based map registration for one-dimensional longitudinal localization. A track-bound coordinate system decouples the lateral and longitudinal track positions. It consists of the longitudinal coordinate s, the lateral coordinate d, and the heading ψ as depicted in Figure 4.7. This coordinate system is known as *Frenét* [318]. As the localization is developed for ground-bound vehicles only, it runs in 2D for the sake of robustness.

The general concept is depicted in Figure 4.8. The fused and transformed longitudinal and lateral localizations are finally fused with the data from the GNSS and the IMU in a KF [319]. This generates the highly frequent localization and state estimation output used in the other modules of the autonomous racing software stack.

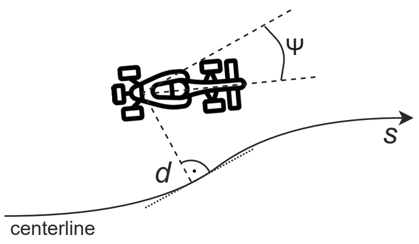


Figure 4.7: Trackbound *Frenét* frame with longitudinal s and lateral d coordinates [11].

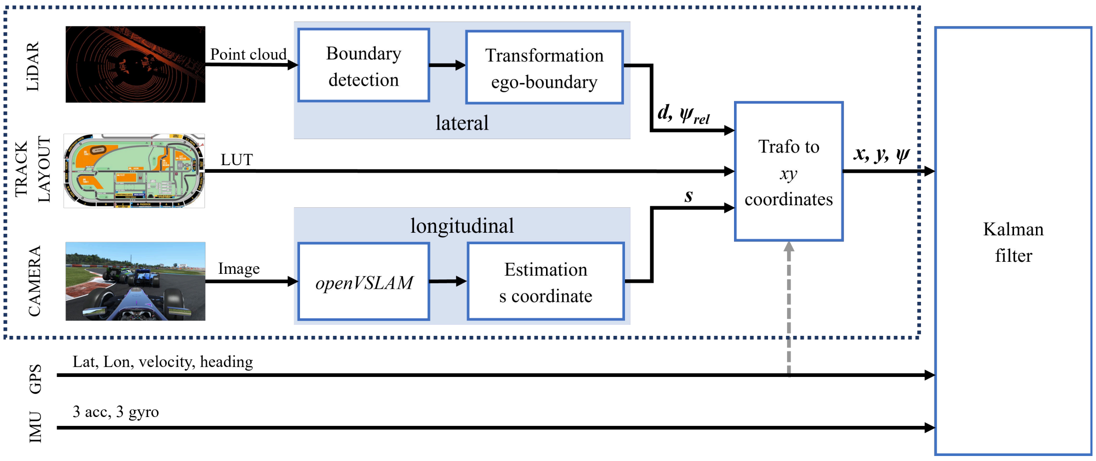


Figure 4.8: Overview of the presented localization approach. Adapted from [11].

Longitudinal Localization

In initial investigations of available visual SLAM approaches, *OpenVSLAM* [125] showed the most promising results. The accuracy and robustness were convincing, and in addition, a map saving and loading functionality was already included. This has some significant advantages: Runtime and compute requirements decrease as no map has to be built online. Furthermore, there is no drift in the position estimate, as the map guarantees global consistency and can be optimized and validated offline. The map is projected into a one-dimensional space along the track to avoid errors in its layout and scale. Each keyframe is assigned its position in the relative longitudinal s coordinate from the start line ($s = 0$) to the finish line ($s = 1$). The total track length is the sum of all the vectors between each of the N keyframes ($\boldsymbol{KFR}$), calculated using the Euclidean norm:

$$tracklength = \sum_{i=0}^{N-1} ||\boldsymbol{KFR}_{i+1} - \boldsymbol{KFR}_i||. \tag{4.1}$$

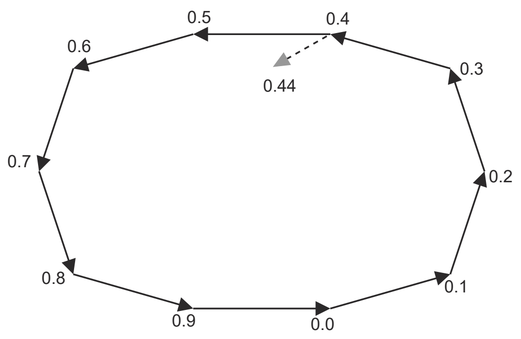
$$s_{KFR_n} = \frac{\sum_{i=0}^{n} ||\boldsymbol{KFR}_{i+1} - \boldsymbol{KFR}_i||}{\sum_{i=0}^{N-1} ||\boldsymbol{KFR}_{i+1} - \boldsymbol{KFR}_i||}. \tag{4.2}$$

The corresponding s coordinate for each keyframe can be determined with the following equation:

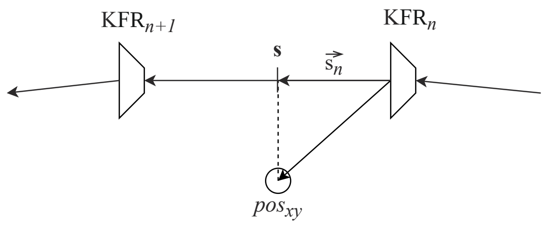
$$\boldsymbol{s}_n = (\boldsymbol{pos}_{xy} - \boldsymbol{KFR}_n) \cdot \frac{\boldsymbol{KFR}_{n+1} - \boldsymbol{KFR}_n}{||\boldsymbol{KFR}_{n+1} - \boldsymbol{KFR}_n||}. \tag{4.3}$$

$$s = s_{KFR_n} + \frac{||\boldsymbol{s}_n||}{||\boldsymbol{KFR}_{n+1} - \boldsymbol{KFR}_n||}. \tag{4.4}$$

Suppose the estimated position of the ego vehicle is between two keyframes. In that case, the s coordinate will be interpolated by taking the orthogonal to the vector between the two closest keyframes. This procedure is shown in Figure 4.9. [320]



(a) Scale-invariant 1D map for longitudinal localization.



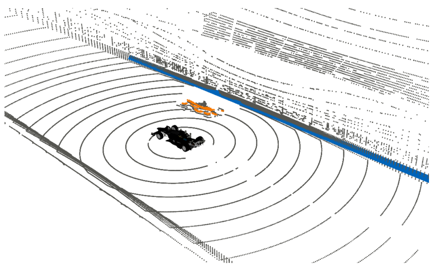
(b) Interpolation of the s coordinate between two keyframes KFR.

Figure 4.9: Transformation of an arbitrary track position into a one-dimensional longitudinal coordinate [11].

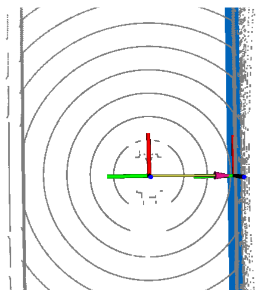
Lateral Localization and Heading Estimation

The second part of the localization pipeline, which runs in parallel, uses the LiDAR sensor and aims to accurately estimate the vehicle's lateral position and relative heading. After applying geometric filtering to the raw point cloud to remove data without relevant information, the SAFER barrier outside the track is detected as shown in Figure 4.10a. The best results for detecting wall points were achieved using basic geometric filtering with fixed assumptions. With this, the ground can be filtered, and only points within a certain height above the ground are kept. Based on these points, the boundary is approximated by a fitted *B-Spline*. This is a piece-wise defined polynomial function that uses the detected points as control points to define the polynomials.

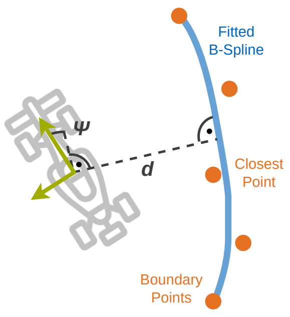
Estimating the vehicle's position and heading relative to the track comprises three steps. First, the closest boundary point is determined as shown in Figure 4.10b. With this distance and the known distance from the boundary to the center line, the d component of the *Frenét* coordinates can be determined in the second step. The third step calculates the angle between the point cloud's x axis, corresponding to the driving direction of the ego vehicle, and the vector pointing towards the closest boundary point. The vehicle heading ψ relative to the boundary can be determined with this angle. The process of the spline fitting and calculation of the ego coordinates is shown in Figure 4.10c.



(a) Detection of the outer track boundary in a $360°$ point cloud. The opponent vehicle is ignored.



(b) Transformation between the ego vehicle and the detected boundary.



(c) Spline fitting and coordinate calculation.

Figure 4.10: Lateral localization via boundary detection, interpolation, and transformation [11].

Since the boundary is assumed to be the farthest right point on the track, point clusters between the ego vehicle and the boundary can be detected. This allows the removal of points that have to come from opposing vehicles or obstacles from the boundary spline fitting. Figure 4.10a shows how an opponent vehicle is detected driving next to the ego (orange). [321]

Sensor Fusion

The prior 2D track information is saved in a lookup table that contains the following information for each data point on the track centerline: s coordinate, its corresponding xy coordinates s_x and s_y, distance to the left boundary, distance to the right boundary, and heading ψ_{global} in global Cartesian xy coordinates. With the estimated s coordinate and the corresponding s_x and s_y, the lateral d coordinate, and the relative heading to the boundary ψ_{local}, the global Cartesian xy pose can be calculated according to Equation (4.5) and Equation (4.6).

$$x = s_x + d\cos(\psi_{\text{global}} + \psi_{\text{local}}). \tag{4.5}$$

$$y = s_y + d\sin(\psi_{\text{global}} + \psi_{\text{local}}). \tag{4.6}$$

Synchronized and triggered frames from LiDAR and the camera are needed for a precise sensor fusion. If the sensors are not triggered simultaneously, the time difference has to be compensated for if simultaneity cannot be assumed. The KF used to further fuse this pose estimate with the data of GNSS and IMU was presented in [319].

Experiment

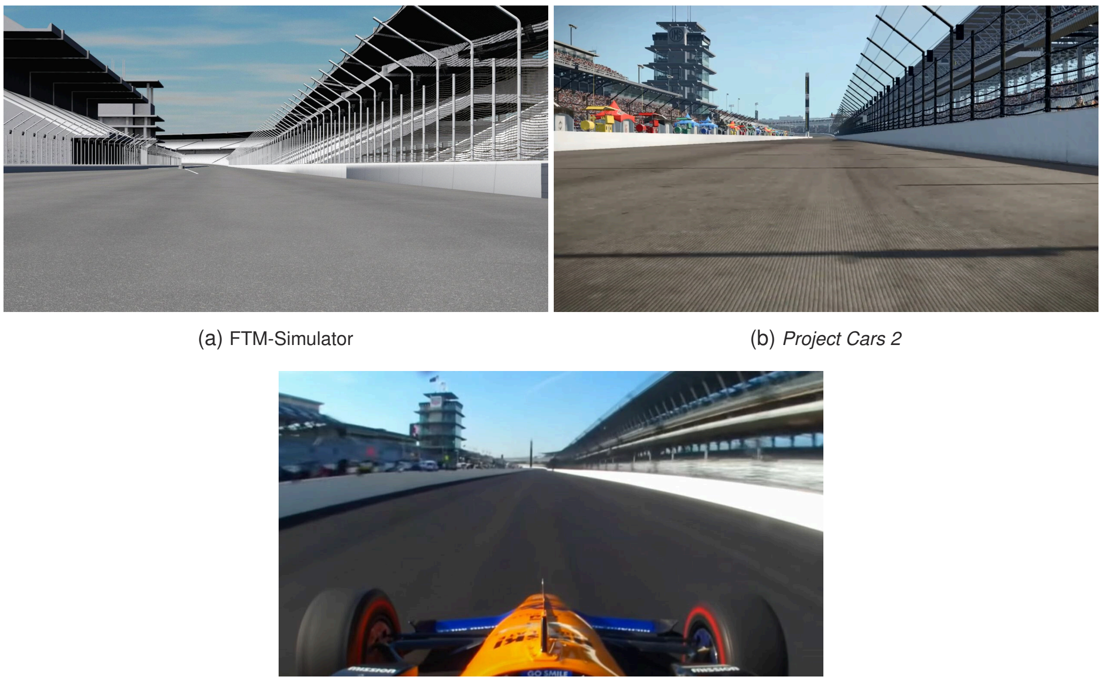
Since this methodology was developed before real-world data from the IAC or the IMS were available or even the sensor suite was defined, synthetic simulated data had to be used for testing and validation. The perception simulation was a custom development (FTM-Simulator) based on the *Unity* game engine. The organizers provided the 3D models of the track, including the surroundings and the vehicle. Synthetic point clouds were generated by a self-developed LiDAR model using ray casting [322]. The FTM-Simulator also opens up the possibility of rendering camera images. Figure 4.11 shows simulated front-camera images from the custom simulator, as well as from the video game *Project Cars 2* and an onboard video from a human-driven Indy car. Mainly because of the more detailed 3D track model, the camera images from the video game appear more realistic. This impression was confirmed by comparing the *OpenVSLAM* on the three types of videos.

To fuse the simulations for the camera and the LiDAR data, the longitudinal localization errors were first determined on the image data created with *Project Cars 2*. Subsequently, these results were fed into the lateral, LiDAR-based pipeline to obtain the final results. By this, the longitudinal error can be determined from the camera simulation data, and later, this information can be used to run the remaining simulation.

Results

For the validation of the longitudinal localization, a feature map was created in a low-speed run. This map is then used for localization during the validation runs. The results were obtained on an *Intel i7-9850H* Central Processing Unit (CPU).

Different velocities with and without opposing racecars are tested on the same map. The average positioning error for runs with speeds between $120\,\mathrm{km\,h^{-1}}$ and $300\,\mathrm{km\,h^{-1}}$ is $0.88\,\mathrm{m}$ with a computation time of around $70\,\mathrm{ms}$. All runs are summarized in Table 4.2.


(a) FTM-Simulator (b) *Project Cars 2*

(c) Formula 1 onboard camera

Figure 4.11: Simulated and real front camera image of the IMS main straight. Right side: SAFER barrier [11].

Table 4.2: Results of the Longitudinal Localization [11].

Longitudinal Deviation						
Velocity	Opponents	Average	RMSE	Maximum	Runtime	Mistracked Frames
$120\,\mathrm{km\,h^{-1}}$	✗	0.557 m	0.843 m	1.347 m	68.9 ms	5
$200\,\mathrm{km\,h^{-1}}$	✗	0.384 m	0.661 m	1.884 m	68.6 ms	2
$200\,\mathrm{km\,h^{-1}}$	✓	0.752 m	1.045 m	2.652 m	67.9 ms	12
$250\,\mathrm{km\,h^{-1}}$	✗	1.326 m	1.851 m	2.163 m	68.4 ms	4
$250\,\mathrm{km\,h^{-1}}$	✓	1.214 m	1.974 m	2.984 m	66.6 ms	7
$300\,\mathrm{km\,h^{-1}}$	✗	0.457 m	1.185 m	3.042 m	70.6 ms	1
$300\,\mathrm{km\,h^{-1}}$	✓	1.471 m	1.992 m	3.692 m	67.3 ms	22

The results of the lateral localization are summarized in Table 4.3. The pipeline based on LiDAR shows more precise results than the camera-based longitudinal localization. The reason for this can be found in the reference used for localization. No distant landmarks are extracted for the LiDAR pipeline, but the SAFER barrier is within a few meters of the sensor. The camera has to use distant features for longitudinal localization. However, due to the overall concept, the longitudinal localization needs to be precise to fully use the potential of the lateral localization.

Figure 4.12 shows the results for the average lateral and longitudinal error of the corresponding algorithm in dependence on the velocity and opponent vehicles. Opponent vehicles show a higher impact on longitudinal localization. This proves the efficiency of the presented algorithm to detect and ignore opponent vehicles. However, at $250\,\mathrm{km\,h^{-1}}$, it can be seen that the localization without opponents shows strikingly negative results. As the algorithm is not deterministic, such behavior can happen for several reasons.

For more detailed results and data at a higher resolution around the track, the reader is referred to [11].

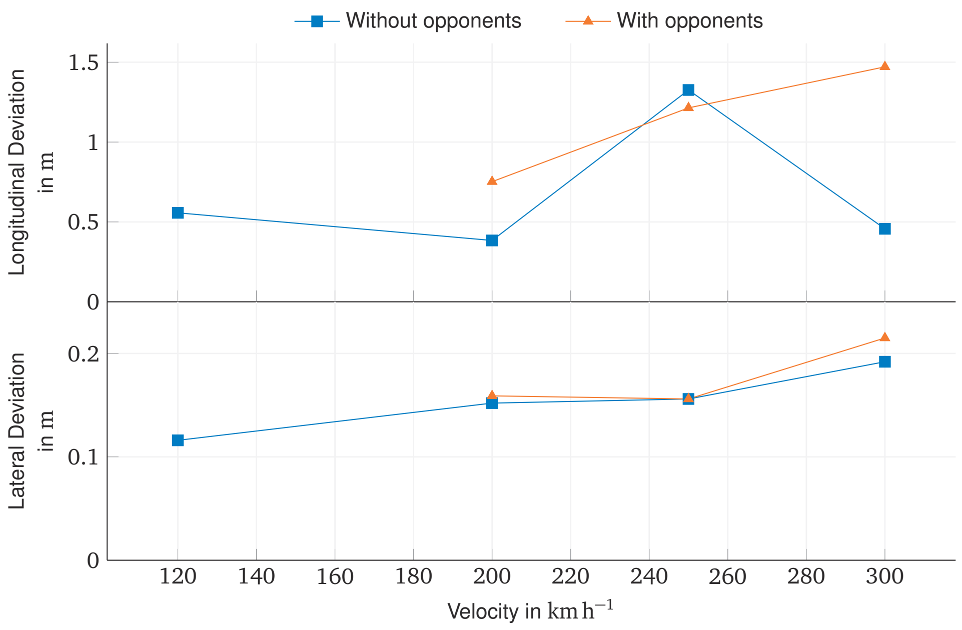


Figure 4.12: Average Deviation of the longitudinal and lateral localization modules.

4.1.4 Results and Discussion

This chapter summarizes the results of the approaches presented in Subsection 4.1.2 and Subsection 4.1.3. Based on this, the methodologies and results are discussed.

The first approach was based on fused point clouds from the three non-overlapping LiDARs. LiDAR-only approaches could not build consistent and accurate maps of oval race tracks. Consistent point cloud maps could only be created using a highly precise GNSS. These maps were shown to be able to support the robustness of the state estimation. However, it was found that scan matching can only cover short-term GNSS outages. High-speed driving in GNSS-denied environments is not possible with the current state. If cm-level accuracy GNSS is available, the addition of point cloud registrations even decreases the precision of the state estimate. The problems of this approach, which hindered better results, were found in several points. A significant challenge and also a difference from most public data sets is the sensor and, specifically, the LiDAR setup of the vehicle. Faulty calibration and missing precise time synchronization cause inconsistencies in each fused 360° point cloud.

Therefore, a novel and integrated approach is proposed for multi-LiDAR fusion and mapping. The structure is depicted in Figure 4.13. This approach will be implemented to the EDGAR research vehicle in Section 5.3. An evaluation of this approach for autonomous racecars on different tracks is currently ongoing and will be

Table 4.3: Results of the Lateral Boundary Localization [11].

		Lateral Deviation			
Velocity	Opponents	Average	RMSE	Maximum	Runtime
120 km h^{-1}	✗	0.116 m	0.171 m	0.732 m	19.4 ms
200 km h^{-1}	✗	0.152 m	0.226 m	0.854 m	19.8 ms
200 km h^{-1}	✓	0.159 m	0.217 m	0.725 m	19.3 ms
250 km h^{-1}	✗	0.156 m	0.244 m	0.931 m	20.1 ms
250 km h^{-1}	✓	0.156 m	0.214 m	0.972 m	19.7 ms
300 km h^{-1}	✗	0.192 m	0.280 m	0.941 m	19.6 ms
300 km h^{-1}	✓	0.215 m	0.401 m	3.116 m	20.2 ms

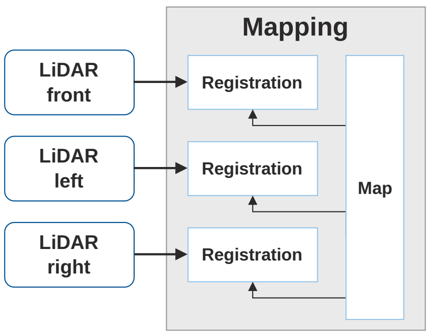


Figure 4.13: Proposed novel LiDARs fusion and mapping approach.

presented in future work. Also, deskewing needs to be performed on the raw point clouds to compensate for motion distortion in the data. Here, a precise state estimation is necessary to extrapolate the position of the points correctly.

Another significant challenge for the LiDAR-based localization and mapping approach is the track surroundings on oval race tracks. Compared to road courses that have been shown to work with LiDAR localization in previous works, oval tracks are inferior in the number of features that can be used for scan matching and localization. The limited range of LiDAR sensors makes it difficult to maintain accurate tracking along these tracks, for example, at the long straights of the IMS. The ongoing development of the IAC will allow comparisons between oval race tracks and road courses in future work.

This section presented a novel approach for the fusion of cameras and LiDARs to overcome these challenges. It uses a visual SLAM-based approach for longitudinal localization and extracts the boundary wall (SAFER barrier) from LiDAR point clouds for lateral localization and heading estimation. This approach showed convincing results at velocities up to $300\,\mathrm{km\,h^{-1}}$ with and without opponent vehicles. One significant advantage of this approach is that if the longitudinal localization becomes unprecise or even loses tracking, it still allows the vehicle to keep a distance from the detected outer wall and avoid crashes. It must be discussed that this approach was only validated with simulated camera images and point clouds. Transfer to the real world has not been carried out. However, as the approaches are not deep learning-based, it can be assumed that they will also work on actual data with adapted parameters. The longitudinal localization has already been proven to be efficient in real video data from human-driven racecars. In future work and to obtain more accurate results, the banking in the curves has to be compensated for the lateral localization. Figure 4.14 illustrates the situation. The LiDAR sensor measures the actual distance to the barrier d_{meas}. As the motion planner needs a 2D position, this lateral coordinate is projected onto the ground plane and becomes d_{proj}. This projection depends on the estimated banking angle θ. If the banking estimation (either offline and map-based or online) is imprecise, the projection will propagate the error, and the 2D position estimate will suffer; thus, the position estimate will be inaccurate.

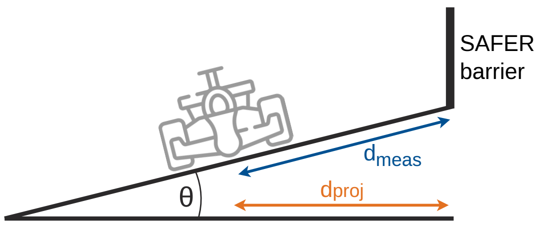


Figure 4.14: Difference between the measured lateral coordinate d_{meas} and d_{proj} in BEV projection.

Synchronized and triggered sensors are assumed in the described experiment, which is not met by the real vehicle. This leads to an additional loss of performance in the sensor fusion module. To reduce this impact, the ego-motion in the time interval between the recording of the camera images and the LiDAR point clouds needs to be compensated for. Only then are the data fused at the correctly extrapolated point on the track.

As the final real racecar had two independent RTK-corrected GNSS systems, these were the primary localization source for the final races. The oval race tracks are open air without any significant obstacles influencing the GNSS signal. Therefore, cm-level accuracy was reached along the whole track. Combined with the EKF-based state estimation fused with IMUs, a precise state estimate could be output at a high frequency. The presented methods are the foundation for future challenges and race tracks that will not allow such precise GNSS localization.

4.2 Depth Enhanced Visual SLAM

The previous section showed how localization and mapping can benefit from sensor fusion, especially in complex environments. To transfer the findings to public roads, this section presents methodologies to enhance existing SLAM approaches with sensor fusion to improve their results. The section tries to find a way to combine the attention to detail of the cameras with depth measurements from LiDAR. Therefore, one of the most famous visual SLAM implementations, the *ORB-SLAM3* [124] is chosen as the baseline and will be enhanced with depth information in Subsection 4.2.1. The following Subsection 4.2.2 will extend this work and aim to find advanced methods for generating dense depth information.

4.2.1 Indirect Visual SLAM with Dense Depth Maps

The content of this section is based on the publication [14], developed and published within the scope of this thesis. The reader is referred to this publication for more in-depth information regarding detailed state of the art, implementation, experiments, and results.
The code is available open-source under https:// github.com/ TUMFTM/ ORB_SLAM3_RGBL.

This work extends the visual SLAM *ORB-SLAM3* [124] by integrating and upsampling depth data from a LiDAR sensor. To efficiently fuse raw data from LiDAR and the camera, the LiDAR points are projected into the camera frame as described in Subsubsection 2.1.2. Therefore, internal camera and external LiDAR-camera calibration matrices are needed. An example of the resulting overlay of depth measurements and camera pixels can be seen in Figure 4.15. The calibration matrices are given by the *KITTI* dataset [42].

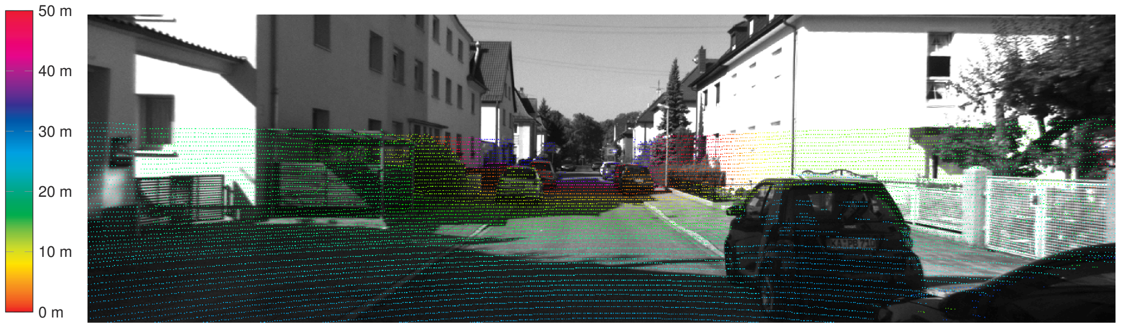


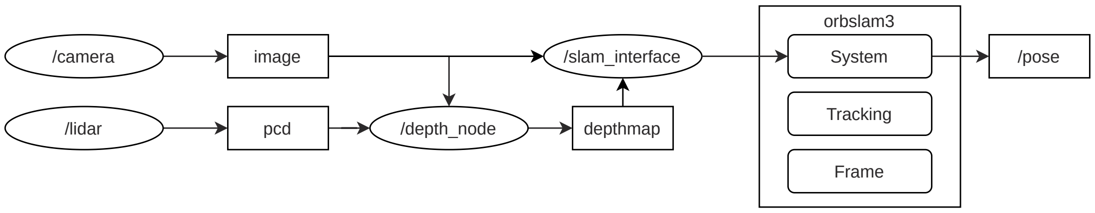
Figure 4.15: LiDAR point cloud projected onto a grayscale image from the *KITTI* Odometry dataset [42]. Each point is colored according to its distance from the sensor.

As this figure reveals, due to the higher spatial resolution of the camera, not every pixel can be assigned a depth value from a LiDAR point. In the *KITTI* dataset [42], the sparsity after projection of the 64-layer

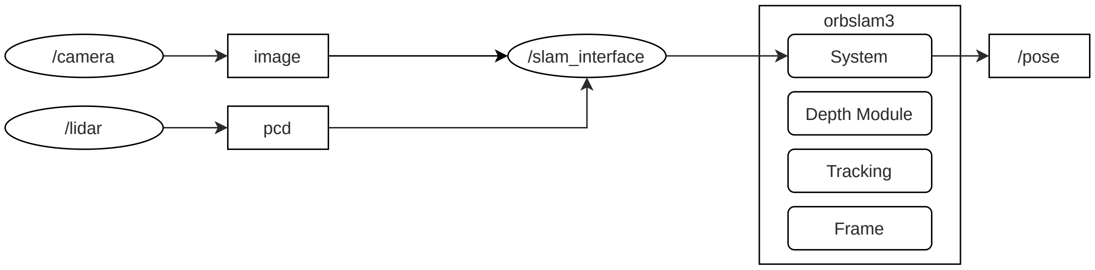
Velodyne LiDAR amounts around $96\,\%$. To assign depth values to pixels between the depth measurements, the depth information needs to be upsampled by interpolation. The presented approach incorporates different LiDAR point cloud upsampling methods directly in the *ORB-SLAM3* [124].

Concept Overview

The *ORB-SLAM3* already incorporates an RGB-D mode, designed to work with RGB-D cameras. In addition to Red, Green, Blue (RGB) images, RGB-D cameras output depth information, usually from stereo or time-of-flight sensors. These cameras typically have a range of a few meters and are mainly suitable for indoor applications [323]. The presented approach builds on the RGB-D mode and extends it by adding LiDAR data. This newly introduced mode is called RGB-L. Two approaches for LiDAR depth integration are implemented and integrated into the *ORB-SLAM3*. The respective computation graphs are depicted in Figure 4.16. The first approach (Figure 4.16a) uses the existing RGB-D mode and externally creates the upsampled depth maps. The second approach (Figure 4.16b) directly integrates the depth upsampling module into the *ORB-SLAM3* core module. Its interface receives the images and point clouds directly; the upsampled depth map is created internally.



(a) Standard *ORB-SLAM3* RGB-D mode with external depth map generation



(b) Novel RGB-L mode with integrated depth module

Figure 4.16: Setup of the *ORB-SLAM3* for working with LiDAR-based dense depth maps in the standard RGB-D mode and the novel RGB-L mode [14].

Depth Map Generation and Upsampling

The depth module aims to first project the LiDAR points into the camera image plane and then to upsample the data to obtain denser depth maps. As sparse depth measurements are available, this problem is generally referred to as depth completion. As described in Subsubsection 2.1.2, depth completion can be performed with conventional CV or deep learning methods.

The presented approach implements both approaches for comparison. The conventional method is implemented in *C++* and can be run as an external module or directly included in the *ORB-SLAM3* as shown in Figure 4.16. Numerous experiments with various conventional upsampling techniques have demonstrated that an inverse dilation with a 5x5 diamond kernel produces the most favorable results among conventional CV techniques [324]. Algorithm 2 represents the kernel dilation upsampling algorithm.

Algorithm 2 Sparse LiDAR Points Projection and Upsampling with Diamond Kernel Dilation

```
    Input: Sparse LiDAR points, camera image
    Output: Upsampled depth map
 1: Initialize empty depth map with size of image
 2: Project sparse LiDAR points into image plane
 3: for each projected LiDAR point do
 4:     Assign depth value to corresponding pixel in depth map
 5: end for
 6: DEFINE 5x5 diamond-shaped kernel with center point (2,2)
 7: for each pixel (i,j) in depth map do
 8:     if pixel (i,j) has a depth value then
 9:         for each position (k,l) in diamond kernel do
10:             Calculate corresponding image pixel (m,n) by offsetting (i,j) with (k,l)
11:             if pixel (m,n) is within image boundaries then
12:                 Assign depth value of pixel (i,j) to pixel (m,n) if it has a higher depth value
13:             end if
14:         end for
15:     end if
16: end for
17: return Depth map
```

A detailed investigation of different conventional CV algorithms and their performances can be found in [324]. The presented method decreases the average sparsity of the depth maps from around $96\,\%$ to around $65\,\%$, leading to a depth map more than five times denser. The resulting depth maps have a Mean Average Error (MAE) of $1.03\,\mathrm{m}$ and a Root Mean Square Error (RMSE) of $4.41\,\mathrm{m}$. Upsampling methods with denser outputs showed worse results in the tracking performance of the *ORB-SLAM3* as the uncertainty increases between the discrete LiDAR points. Deep learning-based methods can predict outputs with a sparsity of $0\,\%$, meaning that every pixel receives a depth value assigned. For this work, the state-of-the-art network *ENet* [325] was chosen for deep depth completion. It achieves an MAE of $0.39\,\mathrm{m}$ and an RMSE of $1.17\,\mathrm{m}$ on the *KITTI Depth* dataset. As for the *KITTI Odometry* benchmark, no depth ground truth values are available; these results were obtained on the *KITTI Depth* dataset. Since no LiDAR depth measurements are available for the upper part of the image (Figure 4.15), the predicted values in this area are not used.

Results and Discussion

For the evaluation of this approach, the *KITTI Odometry* dataset [42] was chosen as it provides ground-truth trajectories, and the appearance of the data is similar to the *KITTI Depth* data, on which deep learning-based depth completion was validated. The presented approaches are compared with the *ORB-SLAM3* stereo mode.

The translational and rotational Relative Pose Errors (RPEs) across the sequences are summarized in Table 4.4. The stereo and RGB-L modes exhibit similar performance levels, with the stereo mode having a marginal edge. A notable observation is that deep learning methods show significantly lower accuracy in both translational and rotational aspects. This might stem from the challenge of defining the general image areas where the network shows convincing results. As described above, the top $30\,\%$ of the depth maps are excluded from consideration to mitigate this. Another limitation of the neural network approach is related to the *KITTI* ground truth, which caps depth values at 80 meters, leading to inaccuracies for objects located further away that are captured by the camera. On the contrary, the RGB-L mode, which is more closely based on actual LiDAR depth measurements, shows better generalization. Detailed analysis across various sequences and sceneries reveals that the RGB-L mode is particularly effective in environments with few distinctive features, namely highway and rural roads. This underlines the theory of the previous section that sensor fusion is especially effective in difficult, feature-poor areas, such as highways or race tracks.

Table 4.4: Average translational error in % and rotational error in °/100 m of *ORB-SLAM3*. Optimum results are highlighted in green, and the next best in blue [14].

Seq.	Scenery	Stereo	RGB-L	Deep Learning	Stereo	RGB-L	Deep Learning
		Translational RPE in %			Rotational RPE in °/100 m		
00	Urban	0.704	0.695	0.721	0.272	0.257	0.315
01	Highway	1.628	1.098	1.698	0.201	0.364	0.681
02	Urban	0.817	0.836	0.852	0.275	0.259	0.344
03	Urban	0.950	1.033	0.726	0.235	0.258	0.334
04	Rural	0.562	0.453	1.121	0.251	0.221	0.985
05	Urban	0.413	0.470	0.840	0.158	0.214	0.419
06	Urban	0.544	0.878	1.182	0.224	0.466	0.425
07	Urban	0.483	0.581	0.862	0.271	0.293	0.497
08	Urban	0.991	1.085	1.284	0.303	0.338	0.556
09	Rural	0.988	0.866	1.092	0.299	0.342	0.438
10	Urban	0.710	0.840	1.388	0.343	0.421	0.484
Average		0.800	0.801	1.070	0.244	0.312	0.498

Another advantage of the novel RGB-L mode becomes obvious when looking at the computational runtimes. The mean runtimes and their variances across all sequences are depicted in Figure 4.17. The two modes that depend on depth maps rather than stereo vision show significantly lower variances. Furthermore, the mean runtime of RGB-L is $22.8\,\mathrm{ms}$ and thus is more than $40\,\%$ reduced compared to $41.2\,\mathrm{ms}$ in stereo mode. It should be noted that the runtimes used for the deep learning-based RGB-D mode are calculated as the sum of the *ENet* and the *ORB-SLAM3* in RGB-D mode. Thus, additional latencies, for example, from the data transmission, can arise in a real runtime application. The presented *ORB-SLAM3* RGB-L mode shows low compute requirements and runtimes. This can make the approach interesting for embedded applications with low-power CPU and without Graphics Processing Unit (GPU).

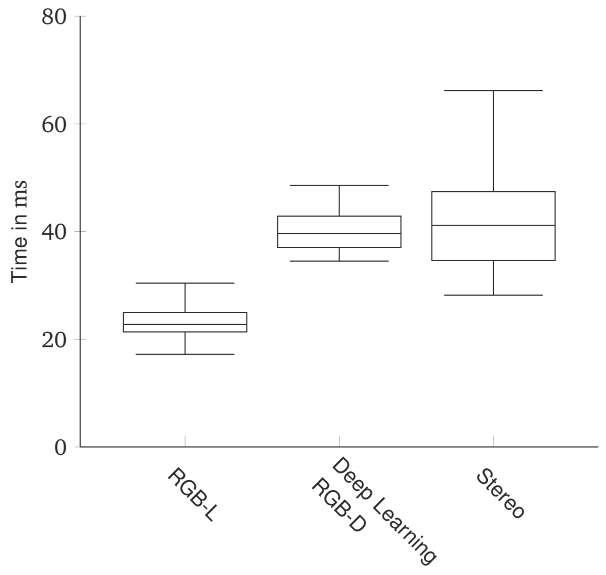


Figure 4.17: Runtime comparison of the presented methods [14].

For more detailed results, deeper investigations on the data used are needed. Therefore, different datasets are to be evaluated. Due to the chosen approach based on conventional CV, this is possible without any retraining. Only the calibration matrices for the internal camera and camera-LiDAR calibration are needed. Calibration is another topic for discussion. In the presented evaluation, the calibration given by the *KITTI* dataset was used. However, investigations in related work revealed a high dependence of visual SLAM on the accuracy of calibration [126, 127]. The public calibration files of the dataset are not perfectly correct and thus induce errors in the depth upsampling. A sensitivity analysis with different modifications of the calibration matrices could give more insight. In addition, the generation of depth maps leaves room for improvement. For example, other input sensor modalities could be evaluated. The following section will present an approach to generate dense depth maps with camera and RaDAR input data. Finally, since this approach has been shown to be especially beneficial for low-feature environments, such as highways, combined with low computation times, this approach can be interesting for autonomous racing applications. This transfer to race tracks is currently under investigation and will be the subject of future research.

4.2.2 Depth Map Generation with Transformers

This section is based on the publication [16]. This article contains additional information on the network layout, experiments, results, and discussion.
The open-source code is available under https:// github.com/ TUMFTM/ CamRaDepth.

As described in the previous section, this approach, called *CamRaDepth* (Camera-RaDAR fusion for Depth estimation), aims to enhance the state of the art in terms of depth estimation. As many existing methods have already exploited the use of expensive LiDAR measurements, this work aims to optimize the output from RaDAR data in combination with 2D camera images. Therefore, a neural network is designed based on the latest progress in the field of ViTs. Compared to LiDAR, RaDAR data are sparse and scarce. To make better use of the data, the network estimates not only a pixel-wise depth map but also a semantic segmentation mask of the image. This additional layer enables a better understanding of objects and semantics and, thus, correlated depth structures.

Data Preprocessing

As there are not many automotive datasets available that provide RaDAR data, the *nuScenes* [45] dataset was chosen for the evaluation and development of this approach. The model takes six feature maps as input: three from the camera image and three from the RaDAR reflection:

- Camera: R, G, B color values
- RaDAR: distance, radial velocity, flow

The data preprocessing pipeline is based on the one presented in [326]. For a more detailed evaluation, the *nuScenes* description text files are parsed to differentiate between day-clear, rainy, and night scenes. This allows for additional evaluation of the contributions of the RaDAR data. The samples are divided into training, validation, and test splits in a ratio of 0.8, 0.1, and 0.1, respectively.

To improve efficiency and runtimes, the **camera image** resolution of the *nuScenes* dataset is modified. The original images, sized at 900×1600, are resized to 416×800 pixels. Similarly to Subsection 4.2.1, the upper part of the image without depth ground truth data is cropped. A reduced image size is chosen for the sake of computational requirements. At the same time, the resolution is still high enough to allow for accurate estimations of depth, especially fine details.

To increase the density of the **RaDAR** points, *superresolution* [199] is used. Therefore, RaDAR frames over a timespan of $0.3\,s$ are accumulated with compensation for ego motion. Furthermore, the RaDAR flow is calculated offline to be used during the training process. This step must be performed at runtime for the application on a real vehicle and live data.

As the *nuScenes* dataset does not provide **depth ground truth**, it has to be calculated from existing sensory data. Also, the semantic segmentation labels needed to train the model's segmentation branch must be created. As LiDAR provides the most precise and robust depth information available, the depth ground truth is based on it. Therefore, the four previous scans and the 21 subsequent scans of the 32-beam LiDAR are accumulated [326]. These accumulated points are projected into the camera image with the given calibration matrices to obtain the associations between the depth values and the camera pixels. The point clouds are motion compensated, and moving objects detected by semantic segmentation are removed to decrease inconsistencies in the data. This depth map is still sparse compared to the camera image, so a dilated depth mask introduces a second ground truth layer in this approach. Therefore, the Algorithm 2 is used that already showed convincing results in the previous approach. To generate **semantic segmentation ground truth**, a state-of-the-art model, namely *mseg* [327] is used. Since the aim of the segmentation branch is only to support depth estimation, the use of existing models for labeling instead of expensive manual labeling was found sufficient.

Model Architecture, Training, and Inference

The main model architecture is based on the *U-net* [328]. Skip-connections are used to embed fine details from the input camera images. The convolutional encoder is replaced by a simplified transformer backbone [329] that has proven superior to convolutional approaches. The complete network architecture is shown in Figure 4.18.

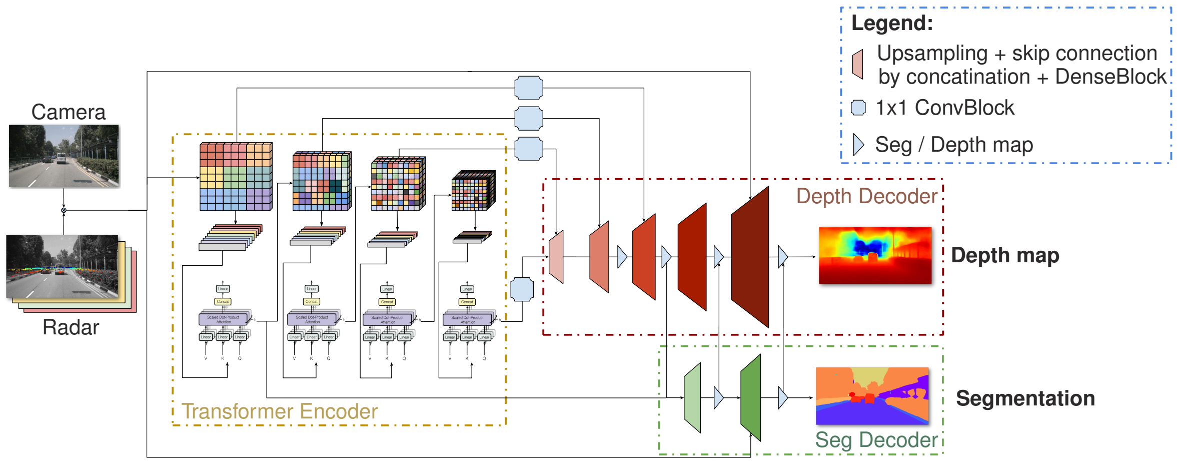


Figure 4.18: Architecture of the introduced *CamRaDepth* network with camera and RaDAR data, and supervised segmentation branch. In later experiments, an unsupervised segmentation branch was added [16].

For efficient and robust learning, the depth values are first normalized to the [0, 1] range. After inference, the model's outputs are then rescaled back to their actual metric values. The encoder consists of four transformer layers to reduce spatial dimensionality. The dimensionality is then upsampled through five steps in the decoder. Therefore, *bicubic* interpolation layers are used. A heavier *dense* block is applied for the final post-processing. From the second upsampling step on, depth calculation blocks are used to generate intermediate depth maps. These allow for intermediate data fusion with the segmentation branch. The depth maps are predicted using a *sigmoid* activation function. The segmentation branch generates two intermediate segmentation masks. The segmentation and depth branch features are merged and put into the depth activation blocks. The network uses group normalization instead of batch normalization, showing superior performance with batch sizes smaller than 16. *GELU* [330] is chosen as activation function. A major problem was found in overfitting during the training process. To avoid this, several countermeasures are included in the network, including dropout layers and *superconvergence* [331].

Different loss functions complement each other to produce accurate, robust, and detailed depth maps. An Mean Squared Error (MSE) reconstruction loss is used for the LiDAR-based ground truth depth map and the denser dilated depth map. The segmentation branch is trained by a *Focal* loss [332]. Finally, this approach introduces a novel loss function called the *Infinity* loss. The idea is to identify the sky regions in the image and push the depth towards infinity. As for these regions with infinitely large depth values, no ground-truth depth can exist, and related approaches struggle in that region.

The depth value of each pixel $\hat{y_{ij}}$ that is located in a region segmented as the sky is to be pushed to zero. As inverse depth maps are used, this relates to an infinitely large depth. For each pixel in the predicted depth map, the loss is defined by

$$\mathcal{L}_{\text{Infinity}}(\hat{y}, y, ft) = (1 - y) \cdot \max(0, \hat{y}) + y \cdot \max(0, ft - \hat{y}) \tag{4.7}$$

where $\hat{y}$ denotes the predicted depth; $y \in 0, 1$ is 0 if the pixel is segmented as sky pixel, 1 otherwise. ft is the true threshold for the sky.

In summary, four loss functions are applied:

- MSE loss between predicted depth map and LiDAR depth ground truth
- MSE loss between predicted depth map and dilated depth ground truth
- *Focal* loss between predicted segmentation and offline generated segmentation ground truth
- *Infinity* loss to push sky pixels to infinite depth

These four functions are weighted with values of 2, 0.75, 0.2, and 0.0005, respectively. The weightings have been determined empirically and fixed, as dynamic weighting decreased the robustness during training.

***CamRaDepth* - Results and Discussion**

The network is trained and validated on the *nuScenes* dataset [45]. By this, it could also be compared with related methods in the field of camera-RaDAR-depth completion. To quantify the results, RMSE, MAE, and Absolute Relative Error (Abs-REL) are used. The Abs-REL calculates the percentage of the prediction error compared to the absolute depth. The interdependencies of the model are investigated utilizing ablation studies. Therefore, transfer learning is a crucial technique so that the entire model is not trained from scratch every time. It encompassed strategically adding or removing layers that needed training from scratch while leveraging pre-converged layers to significantly speed up their training process. The initial model is trained entirely from scratch, comprising three distinct branches: one dedicated to depth estimation and the others to supervised and unsupervised semantic segmentation. After this initial model has reached convergence, ablation studies are performed by removing one branch at a time. Each new iteration of the model is initialized with the previously converged weights, facilitating a more rapid training process and leading to promising results with greater efficiency. In total, six different evaluations are performed:

(1) Base model only with RGB input, no RaDAR

(2) Base model with RGB and RaDAR, no semantic segmentation

(3) Model with RGB, RaDAR, and unsupervised semantic segmentation

(4) Model with RGB, RaDAR, and supervised semantic segmentation

(5) Model with RGB, RaDAR, unsupervised, and supervised semantic segmentation

(6) Base model with RGB and RaDAR, no semantic segmentation, trained from scratch

(7) Model with all branches, trained with data augmentation

All models are trained and executed using a *NVIDIA V-100* GPU with 16 GB of VRAM. The total training consists of 60 epochs and 20 fine-tuning epochs. Full training takes around 75 hours in total. Although the models differ in number of branches and parameters, they all have an inference time of $\sim 55\,\mathrm{ms}$ per iteration. Memory consumption on the GPU during inference amounts to around 1.5 GB. Model (7) is trained using transfer learning as the initial weights are taken from the converged model (5). During the training process, data augmentation is applied to obtain more robust results and better generalization. Therefore, the RaDAR depth maps and the ones from LiDAR ground truth were swapped randomly, although their distributions differ considerably. This swap was done with a probability of $p = 0.18$ to avoid overfitting. The largest model (5) has a total of 23 million parameters. All results from the described experiments are summarized in Table 4.5. It also contains the results of the baseline implementation, a purely convolutional implementation [333], and approaches from related work.

Table 4.5: Comparison of the presented method with others, evaluated on *nuScenes*. The difference in performance between the different variants of the model is negligible. It is predominantly influenced by two factors: the number of training steps used in fine-tuning and the state of initial weights when applying transfer learning. This can be observed by looking at (5), the first to train, and (3), the last. The importance of semantic segmentation guidance is highlighted in (6), compared to (1), which was trained using guided weights. The base model falls significantly behind when trained from scratch. [16]

	Metric	Input	Resolution	Max Depth	*Abs-REL* ↓	*MAE* ↓	*RMSE (in* m*)* ↓	
	Conditions				MIXED		MIXED	MIXED (50 m)
SOTA	RC-PDA [326] (results from [334])	RGB + Radar	192×400	80 m	0.128	-	6.942	-
	RC-PDA [326]	RGB + Radar	192×400	50 m	(0.085)	(1.655)	-	3.179
	Lin et al. [335]	RGB + Radar	450×800	-	0.100	2.061	5.180	-
	RC-DPT [334]	RGB + Radar	320×576	80 m	0.095	-	5.165	-
	Lee et al. [336]	RGB + Radar	450×800	70 m	0.104	2.104	5.209	-
	Singh et al. [337]	RGB + Radar	900×1600	50 m	-	(1.728)	-	3.747
	Singh et al. [337]	RGB + Radar	900×1600	70 m	-	2.073	4.591	-
	Singh et al. [337]	RGB + Radar	900×1600	80 m	-	2.179	4.899	-
	Convolution + Atten + Seg [333]	RGB + Radar	416×800	100 m	0.127	2.924	5.442	3.869
OURS	(1) Transformer w/o Seg	RGB	416×800	100 m	0.087	1.971	4.008	3.156
	(2) Transformer w/o Seg	RGB + Radar	416×800	100 m	0.088	1.963	3.987	3.152
	(3) Transformer w/ Seg (Unsup.)	RGB + Radar	416×800	100 m	0.088	1.945	3.963	3.133
	(4) Transformer w/ Seg (Sup.)	RGB + Radar	416×800	100 m	0.090	2.018	4.014	3.179
	(5) Transformer w/ Seg (both)	RGB + Radar	416×800	100 m	0.091	2.050	4.043	3.193
	(6) Transformer (from scratch)	RGB + Radar	416×800	100 m	0.096	2.197	4.362	3.482
	(7) Transformer with data aug.	RGB + Radar	416×800	100 m	**0.072**	**1.705**	**3.649**	**2.773**

Figure 4.19 shows exemplary inputs and outputs of the network. One can see how the segmentation branch reconstructs the semantic segmentation, whereas the unsupervised segmentation branch interprets the image structure differently. Compared to the baseline model, the most significant performance boost was the adaptation of ViT as an encoder. The base model trained from scratch could already reach an RMSE of $4.2\,\mathrm{m}$ and was on a level with related publications. Also, the specific training method that uses *superconvergence* [331] and cyclic learning rate scheduling helped overcome the higher errors the network tended to converge.

The significant impact and benefit of transfer learning can be seen when comparing models (2) and (6). Even though they share the same network architecture, model (6) performs around $8.5\,\%$ worse in terms of RMSE. The only difference is that it was trained from scratch. This means that even the base model can benefit from already converged weights that have been trained with an additional segmentation branch. The network does not need to perform the segmentation during inference. This can be seen by the negligible performance benefit of the models with segmentation branches at inference. However, segmentation branches are of benefit during the training process. The lack of generating detailed depth maps without segmentation can be visually determined in Figure 4.20. Here, the model without segmentation, trained from scratch, fails to

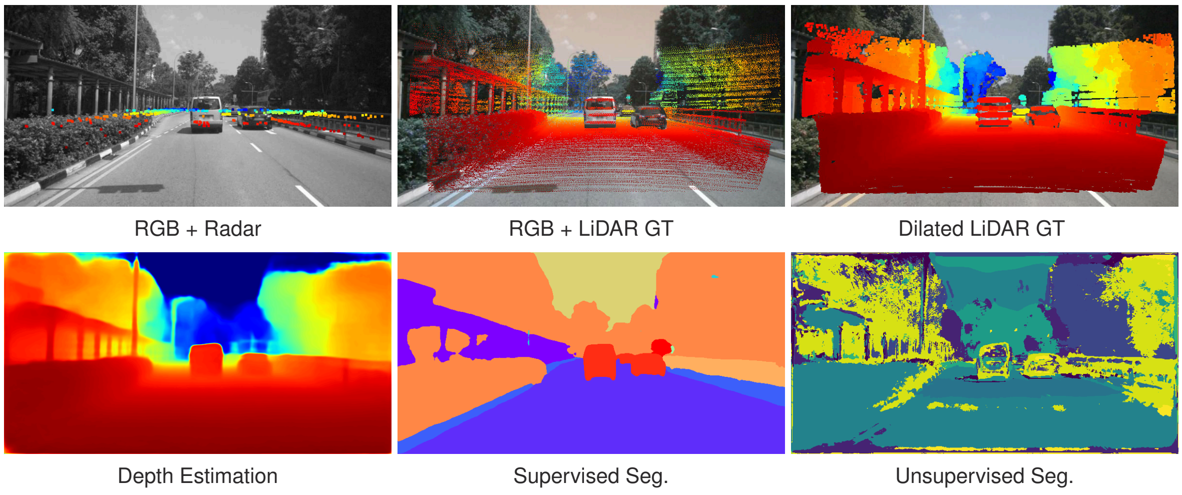


Figure 4.19: Inputs and outputs of the model. While the supervised segmentation model gives the most realistic-looking depth map, the unsupervised segmentation model shows capabilities of detecting both fine details and surfaces. [16]

reconstruct the details. The model without segmentation trained via transfer learning behaves similarly to the one also running the segmentation during inference. Figure 4.20 also reveils the benefit of the novel *Infinity* loss. The bottom right image shows how the model trained with additional *Infinity* loss successfully segments the sky and asserts high depth values. However, the corner of the building (B) also shows that the sky segmentation is imperfect.

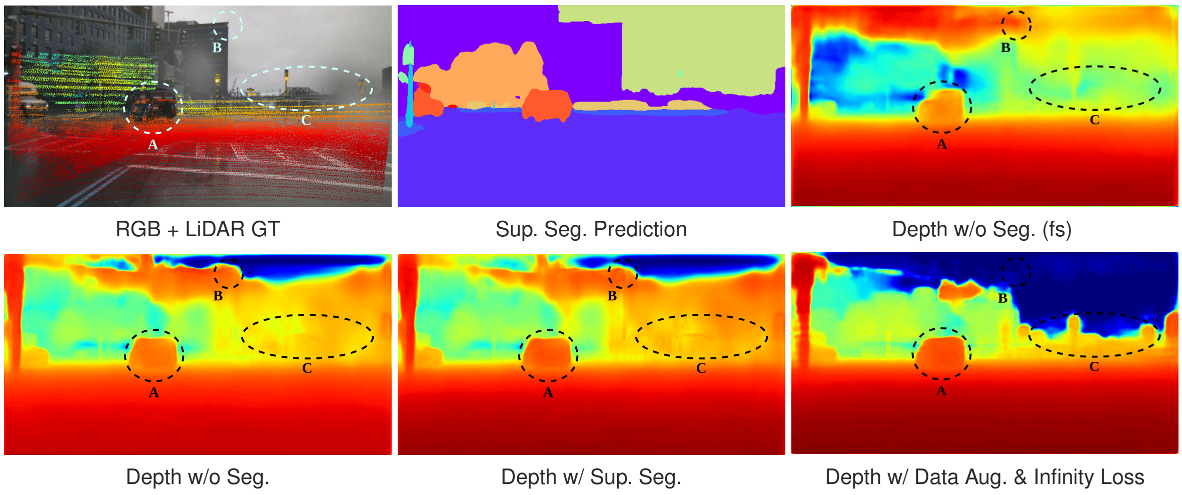


Figure 4.20: A rainy scene. This example shows how additional information from the segmentation branch adds details to vast areas of uncertainty. Also, the effect of the *Infinity* loss for the sky can be seen. [16]

As Table 4.5 shows, another remarkable performance gain is achieved by adding the previously described data augmentation scheme achieves another remarkable performance gain. Therefore, the model is first fully trained with the standard RaDAR data and then fine-tuned, randomly ($p = 0.18$) replacing the RaDAR data with LiDAR points. Applying this augmentation scheme from the initial training steps resulted in overfitting behavior. The supervised semantic segmentation branch enables the model to perceive the depth of small details; however, it sometimes enforces constant depth values at surfaces with degrading depth. Therefore, an unsupervised segmentation branch is introduced that manages to find a balance between. The optimal configuration of this unsupervised branch will have to be investigated in future research.

Another point of discussion is the quality of the ground truth. As no ground truth is provided for the *nuScenes* dataset and it is generally difficult to obtain ground truth data, it had to be calculated from LiDAR data. However, this approach also has its disadvantages which will be elaborated in the following. Ego motion compensation and removal of dynamic objects have been performed on the data, but this adds uncertainty as the corresponding methods are not error-free. The calibration was taken from the dataset and also shows shortcomings. Figure 4.21 shows how the correspondences between the camera image and the LiDAR points are erroneous (B). Using data augmentation, an additional dilated LiDAR depth map and the *Infinity* loss helped to contain the effects of this, but the problem with the ground truth data remains.

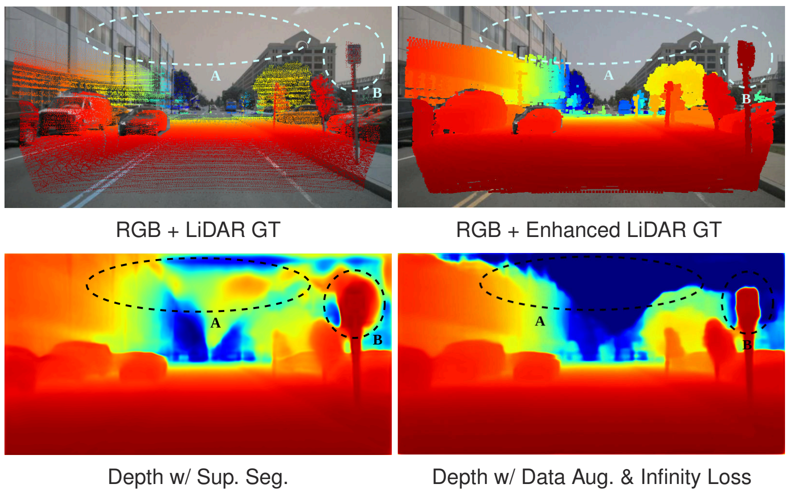


Figure 4.21: Shortcomings of the ground truth. The lack of comprehensive depth ground-truth data in different areas of interest leads to false predictions around isolated objects. **A** demonstrates the struggle to estimate accurate distances for the infinitely far-away sky. **B** is an example of an “island” of uncertainty. A slight miscalibration can be seen in the ground-truth image. [16]

Another problem comes from the limited FOV and range of the LiDAR taken to generate the ground truth. As there are no ground-truth data for some regions of the image, the behavior of the network is undefined. Once again, data augmentation and especially the *Infinity* loss helped to reduce the impact of this (Figure 4.21 A). Due to the limited range of the ground truth, the range advantage of the RaDAR can not be quantified. Also, the advantages of the RaDAR in bad weather cannot be used, as the model will still try to reconstruct the LiDAR measurements.

In summary, it has to be stated that even if the training aims at the lowest error compared to depth maps from LiDAR data, this is not desirable as this means also reconstructing the contained errors. To avoid this, novel methods for ground truth generation will have to be evaluated. Another possibility is to use unsupervised approaches. These use camera images from different perspectives to determine the desired depth maps. Related work shows that these approaches preserve many details but are still inferior to supervised approaches [36]. One possibility of combining the advantages of both methodologies could be hybrid approaches.

More details on the results, especially in bad weather conditions, and a more detailed discussion of the results are given in [16].

4.2.3 Results and Discussion

This section summarizes the results of Subsection 4.2.1 and Subsection 4.2.2. A discussion with a short outlook for future research topics will be derived from this.

The *ORB-SLAM3 RGB-L* [14] integrates LiDAR depth measurements directly into the well-known and frequently used visual SLAM approach. To obtain the associations between the camera pixels and LiDAR reflections, the comparably sparse LiDAR data had to be upsampled. This was done using both conventional CV methods and deep learning methods. Surprisingly, the approach based on conventional CV outperformed deep learning-based depth completion methods regarding the accuracy of SLAM. The accuracy of the stereo mode could be matched and showed robust results, even outperforming it in feature-poor environments such as highways. An additional benefit was the significantly decreased runtime. Due to the lightweight depth-upsampling implementation, it could be reduced by around 40 %. Therefore, this approach demonstrated that the fusion of LiDAR data can improve visual SLAM even with simple data fusion approaches. Here, it should be discussed that deep learning-based depth completion using *ENet* [325] was validated on the *KITTI* Depth [42] data. However, the SLAM results were generated using the *KITTI* odometry data. Therefore, slight differences in data, calibration, etc., can lead to significant differences in data-driven approaches. This fact urges the need for more complete datasets for localization and mapping.

Also, the neural network outputs depth estimations for every single pixel, even if the uncertainty is high. This can be seen in the upper regions of the images, where no LiDAR points are available. As there are no measurements, there is also no deviation from ground truth that can be penalized in the training process. This leads to random behavior in these regions. Similar effects can appear in other image regions between the LiDAR depth measurements. The dilation approach has the advantage that it only outputs depth estimations in the neighborhood of measurements. This leads to more manageable uncertainties.

A general challenge of deep learning-based depth estimation is the ground-truth data used. For the approaches presented, the depth ground truth was generated by accumulating motion-compensated LiDAR scans. To remove errors, the regions segmented as dynamic objects were removed. As the results of [16] already showed, many problems arise from errors in the ground truth. The capability of LiDAR sensors is limited, and thus, all LiDAR-based ground-truth data are not a real ground truth. Consequently, a neural network that reconstructs these data exactly is not desirable. However, this is what it is trained for during the training process. A solution for this problem can be self-supervised approaches. These generate depth data based on images and ego-pose data during training. This also allows the network to reconstruct fine details that are too fine to be properly captured by LiDARs [16]. However, self-supervised approaches are inferior to supervised, trained networks in terms of performance. As no real ground truth is available, this comparison will also have to be the subject of future investigations.

The increased requirements must be considered a general disadvantage of sensor fusion-based localization approaches and the presented depth enhanced visual SLAM. Certain assumptions are made in the sensor fusion, and the sensor setup needs to fulfill them. First, precise time synchronization (for example, Precision Time Protocol (PTP) or Network Time Protocol (NTP)) is essential for the relation of the sensor data's timestamps to be known. Second, many approaches assume simultaneously triggered sensors. As many sensors do not record all data points simultaneously (e.g., camera shutter or rotating LiDARs), the data must be de-skewed to remove the induced motion blur. If sensors are not triggered simultaneously, their data must be compensated for the difference in time. With known ego movement and a static environment, this is possible. However, ego localization induces uncertainty, and dynamic surrounding objects cause errors in the data as they cannot be compensated for. These problems also exist in monomodal SLAM but do not propagate between sensor modalities.

The previous section's second approach called *CamRaDepth*, aimed to generate robust depth maps even without using expensive LiDAR sensors at runtime. However, since no ground-truth depth values are available, the ground truth was generated based on the accumulation of LiDAR points. The approach showed how well neural networks, especially transformer architectures, can “memorize” the capabilities of additional sensors used during training. This can also be seen when comparing the results with and without RaDAR during inference. There is almost no difference in accuracy. However, the performance was significantly lower when the model was trained from scratch, without seeing any data from RaDARs. Similar results could be witnessed by looking at the semantic segmentation branch. Nevertheless, the concept of multi-task learning was also proved. The approach showed that an additional semantic segmentation branch helps the depth branch to reconstruct fine details. Future research will have to dig deeper into this topic to evaluate if the depth branch can support semantic segmentation vice versa. Finally, incorporating self-supervised approaches, perhaps in a hybrid way in combination with supervision, should be evaluated. They can help generate more consistent ground-truth data.

The approach presented to incorporate depth data into visual SLAM was based on the existing *ORB-SLAM3* [124]. Consequently, this implied some framework conditions. For future implementations, a more integrated approach is desirable. This could also support the quality of depth estimation, as it could directly use the information obtained from SLAM. Similarly to the *CamRaDepth* approach, different downstream tasks (SLAM and depth estimation) using the same input data could benefit from each other. The following section presents a concept for a more integrated SLAM approach.

4.3 CaLiMO - Tightly Coupled Camera LiDAR SLAM

As a novel approach with a more tightly coupled sensor fusion, this section introduces *Camera LiDAR Mapping and Odometry (CaLiMO)*. The tight sensor fusion approach allows for the use of a higher information content of the sensor data. However, as already pointed out, this leads to high requirements for the sensor setup and can also make the approach more sensitive to imperfections in the data sources.

4.3.1 CaLiMO Pipeline

Figure 4.22 shows the structure of the approach. In total, the algorithm is composed of four working threads:

- **Tracking thread** assigns depth from LiDAR to the detected visual GFTT features, runs Lucas-Kanade (LK) flow tracking and detects keyframes.
- **LiDAR odometry** performs scan matching between the LiDAR point clouds corresponding the detected keyframes.
- **Local optimization** is the graph-based back-end of the algorithm and optimizes the poses of all keyframes and landmarks from the tracking thread.
- **Loop closing thread** consistently searches for loops in the trajectory and corrects the loop error.

Subsequently, the components and the mentioned threads are presented and explained.

Tracking

The tracking thread is part of the front-end of *CaLiMO*. Therefore, it creates constraints for the graph optimization by extracting and matching visual features, associating LiDAR depth information, and finding keyframes.

Depth Association:

For the depth association, similar to Subsection 4.2.1, the LiDAR points are first projected into the image space. It is essential for this approach to synchronize the frames of the LiDAR and the camera, ensuring that the measurements from both sensors are temporally aligned. For accurate spatial alignment, precise

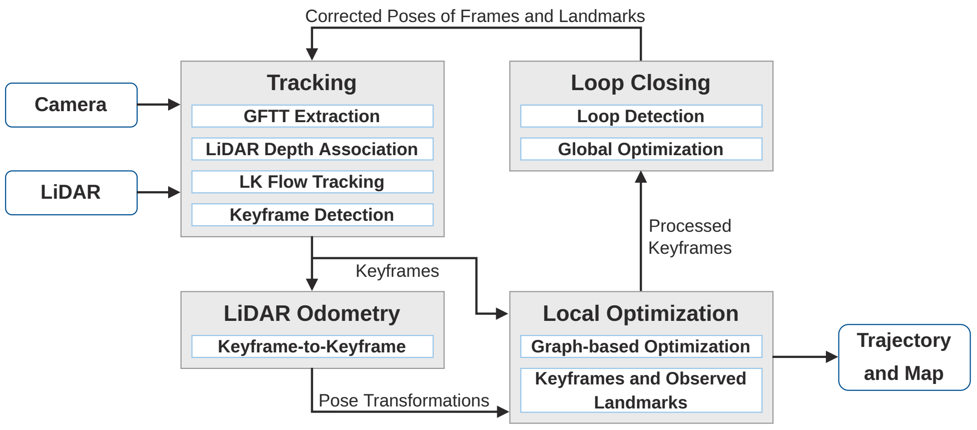


Figure 4.22: Structure of *CaLiMO* corresponding to the threads in the implementation.

external and internal calibration of the sensors is crucial. The technique identifies the nearest LiDAR points to the features of the image and constructs a local 3D plane for depth estimation. A *k-d tree* and the Fast Library for Approximate Nearest Neighbors (FLANN) algorithm [338] facilitate the selection of these points based on proximity within a predefined radius. The method prioritizes vertical coverage due to the sparser vertical resolution of the LiDAR. However, features with wrong depth associations can appear and must be filtered out in the tracking algorithm.

A visualization of a camera image with detected GFTT features and their projection into the 3D LiDAR point cloud is shown in Figure 4.23.

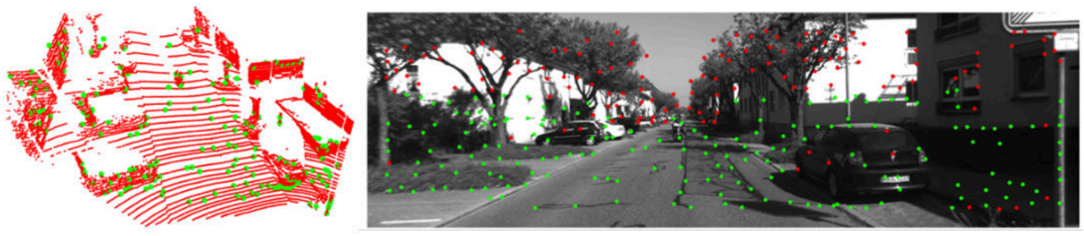


Figure 4.23: Depth association of GFTT features from LiDAR data. Green points are tracked, red points are untracked detected features.

Visual Tracking:

The visual tracking is based on the LK-flow algorithm [339], tracking a sparse set of pixels from frame to frame. The selection of these pixels is typically based on the identification of key corner points in the images. Therefore, it is assumed that the brightness of each pixel remains consistent in subsequent frames. A GFTT feature detector is employed to optimize tracking performance. The GFTT detector is preferred for its accuracy in detecting corner points for optical flow tracking, although it is computationally more demanding than other detectors. The replacement with different feature detectors can be subject to future research. Although the LK-flow method is efficient, it has limitations in dealing with changes in illumination and motion smoothness. To address these, an image pyramid approach is adopted. Tracking starts at the highest level (e.g., smallest image) and progresses to lower levels, reducing the impact of significant movements or illumination changes. An additional outlier removal method is applied. The result is a reliable set of features for subsequent camera pose computation, validating the effectiveness of the proposed method. [340]

LiDAR Odometry

Additional LiDAR scan registration is applied between the keyframes for additional stability of the approach. Some preprocessing steps are performed to apply efficient scan-matching LiDAR odometry. First, obvious outliers that are out of range or out of FOV are removed by geometric filtering. Then, a voxel-grid filter is applied to reduce the amount of data. There, a compromise has to be found between preserving fine features and reducing the amount of data.

In the next step, an NDT algorithm is used to obtain the transformation between the current and the previous keyframes. The initial pose estimation is given by the visual tracking algorithm described above. Even if NDT does not produce valid results in the required runtime, the back-end will continue to optimize and output results without the use of LiDAR odometry.

Graph Optimization

A graph-based back-end is implemented for a precise reconstruction of the pose and trajectory of the ego. Motion-only bundle adjustment enables frame-to-frame tracking, which is required by the front-end of the system. The camera pose can be computed by minimizing the reprojection errors of all observed landmarks in the current frame.

The graph-based local optimization is crucial and required by the back-end. The main objective is to select relevant vertices (e.g., keyframes and landmarks) and edges (e.g., measurements or odometry) from the built graph and jointly optimize them at a lower frequency. The number of keyframes in the local optimization window is set to eight to limit the problem and keep it solvable in real-time. Figure 4.24 depicts the algorithm and how the graph is constructed. A LK-flow tracking is applied between the camera frames, and keyframes are identified. For the GFTT matching between frames, the depth is associated with the corresponding LiDAR point clouds. Additionally, the point clouds corresponding to the keyframes are matched using an NDT approach.

Loop Closing

Camera images are used to detect loop closure. If a loop closure event is detected, the loop closing optimization mainly tries to distribute the loop error over the entire trajectory and thus correct the loop error globally. This thread does not have to run in real-time and can run at a slow frequency in the background. The whole back-end and graph optimization was implemented using the *g2o* library[2]. [340]

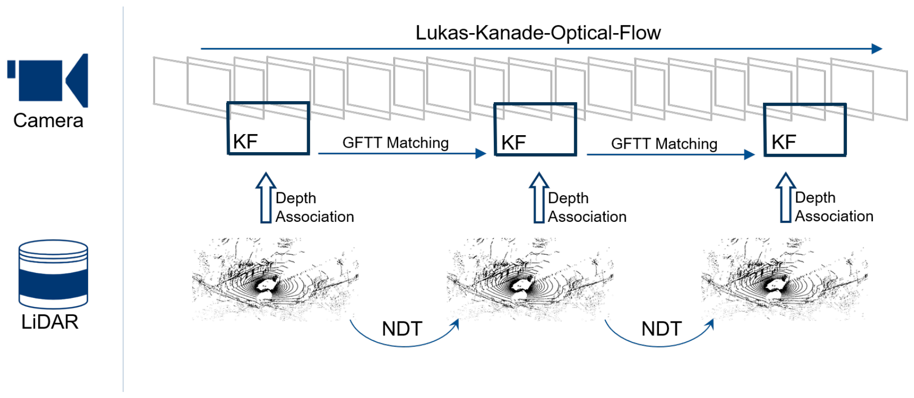


Figure 4.24: Working principle of the *CaLiMO* pipeline.

[2]https://github.com/RainerKuemmerle/g2o

4.3.2 Results and Discussion

To keep the results comparable, the *KITTI* odometry dataset [42] was chosen. It provides synchronized and triggered LiDAR scans, camera images, and corresponding ground-truth trajectories. The results are summarized in Table 4.6. Due to the implementation in *C++*, the algorithm runs in real-time on a consumer processor (*Intel Core i7-9750h*).

Table 4.6: Average translational error in % and rotational error in $^{\circ}/100\,\mathrm{m}$ of *CaLiMO* on the *KITTI* odometry benchmark [42].

Seq.	Scenery	Translational RPE in %	Rotational RPE in $^{\circ}/100\,\mathrm{m}$
00	Urban	0.892	0.327
01	Highway	lost tracking	lost tracking
02	Urban	0.967	0.363
03	Urban	0.839	0.457
04	Rural	1.766	1.465
05	Urban	0.805	0.312
06	Urban	0.913	0.445
07	Urban	0.739	0.328
08	Urban	1.448	0.449
09	Rural	0.832	0.267
10	Urban	0.900	0.561
Average		0.956	0.497

It becomes evident that, in contrast to the *ORB-SLAM3 RGB-L*, *CaLiMO* fails to reconstruct the trajectory from the highway sequence 02. In this sequence, a smaller number of features is found, which leads to challenges for the visual tracking thread. Furthermore, due to the repetitive highway scenario, also the LiDAR odometry cannot support the tracking sufficiently. Here, the more compute-intensive *ORB* features are beneficial. In the future, these could also be integrated into the *CaLiMO* approach. Besides this sequence, the algorithm manages to estimate valid trajectories for all sequences in rural and urban environments. The presented results were obtained without loop correction. The loop closing thread managed to detect loop occurrence events and correct the error in some cases. However, the average results suffered from loop closure addition. Additional investigations are needed to solve this problem.

A big topic for improvement is the implementation and the resulting concurrency of the threads. This appears when loop closure, and thus, global optimization is enabled and is executed in parallel to the local optimization thread. In the current *CaLiMO* implementation, concurrency leads to indeterministic behavior, which is not the case if only locally optimized trajectories are output. For future research, a more stable implementation and some refactoring of the project are needed. Therefore, the current approach needs to be restructured to avoid these issues. However, even without global optimization, the approach showed convincing results and an interesting base for future work. The algorithm must be further enhanced, fine-tuned, and optimized to reach state-of-the-art results.

4.4 Summary of the Results

After extensive exploration of several algorithms for sensor fusion, localization, and mapping in different environments and use cases, autonomous racing and public traffic scenarios, this section aims to summarize the key results and findings of Section 4.1, Section 4.2, and Section 4.3. The primary challenges encountered in the localization approaches presented can be traced back to two root causes: the complexity of the environments and limitations stemming from the sensor setups. These challenges manifested themselves in various forms in the different methodologies and scenarios investigated.

Complex Environments: One of the most significant obstacles was the variability and unpredictability of different environments. Whether it was the high-speed dynamics of a race track or the repeating and monotonous appearance of highways, each environment presented its unique set of challenges. These complexities often tested the limits of algorithms and highlighted the need for robust and adaptable solutions capable of performing under a wide range of conditions.

Sensor Limitations and Calibration: Another critical aspect that influenced the efficacy of the localization techniques was the quality and calibration of the sensors used. Imperfect sensor setups, including problems with temporal synchronization, spatial calibration accuracy, and data alignment, significantly affected system performance. This finding underscores the importance of high-quality sensor data and precise calibration or robust algorithms capable of handling such problems for achieving reliable and accurate localization and mapping.

As all existing approaches of LiDAR localization and mapping failed for the use case of autonomous racing, a novel approach was introduced in Subsection 4.1.3. It uses a camera for longitudinal and a LiDAR sensor for lateral localization relying on pre-built maps. The algorithm was tested and found to be efficient in simulation. However, due to the high requirements for sensor data and the tight timeline of the autonomous racing challenge, the approach could never be validated in real-world applications.

A more general approach was developed to bring the idea of using fine-grained details from the camera image with depth information from LiDAR from race tracks to public roads: *ORB-SLAM3 RGB-L*. It upsamples depth information from LiDAR point clouds and projects them into the corresponding camera frame using the calibration data. Even with a lightweight conventional CV approach to upsampling, using a 5x5 pixel-sized diamond and dilation, the approach showed precision comparable to that of the *ORB-SLAM3* in stereo mode. In the difficult environments described, especially on highways, the *RGB-L* even outperformed the stereo mode and showed to be more robust. Furthermore, the computation time could be reduced by around 40 %.

Subsequently, Subsection 4.2.2 investigated the pixel-wise depth estimation in more detail. Therefore, *CamRaDepth* was developed using camera images and RaDAR measurements. The RaDAR reflections are projected into the camera image, similar to the LiDAR points in the previous approach. It incorporates a ViT encoder and a convolutional decoder. To improve the network's ability to correctly estimate fine details, an additional semantic segmentation branch was introduced for enhanced object-level understanding. The approach showed to be efficient and outperformed the state of the art using the *nuScenes* dataset. Evaluation and several ablation studies showed that the accuracy is insignificantly lower than with the full model even when RaDAR data and the segmentation branch are not available at inference. However, during training, the network benefits heavily from them. This leads to the conclusion that the *CamRaDepth* network manages to save this information in the weights and “memorize” the effects during inference. Although deep learning is an efficient tool for depth estimation, the *ORB-SLAM3 RGB-L* showed the best results with basic dilation upsampling of LiDAR points for semi-dense depth maps. More data are needed for a more detailed investigation of combining the approach with deep learning-based upsampling.

The final section of this chapter introduced *CaLiMO*. This approach aims for a tightly coupled visual LiDAR SLAM. On the *KITTI* dataset, the approach showed meaningful results, managing to keep track of the ego-pose throughout 10 out of 11 sequences. Nevertheless, the results fell short of the results previously shown of *ORB-SLAM3 RGB-L*. It should be mentioned that *CaLiMO* in the current state struggles with global optimization. Furthermore, more resources need to be put into the further development and parametrization of this approach.

In conclusion, while the research in this thesis made significant progress in advancing sensor fusion for localization and mapping, it also highlighted the critical need for further development in dealing with different environments and imperfect sensor setups. These insights pave the way for future research that focuses on improving the robustness of localization algorithms to better adapt to the diverse challenges presented by real-world scenarios.

The next chapter will present a mapping pipeline for an autonomous multisensor research vehicle. All of the presented findings will be used to realize and apply this pipeline.

5 Open Mapping Framework for Urban Environments

Parts of this chapter appeared in [18].

This chapter mainly addresses the second research subquestion:

SQ2: How to design an open HD mapping pipeline for AVs?

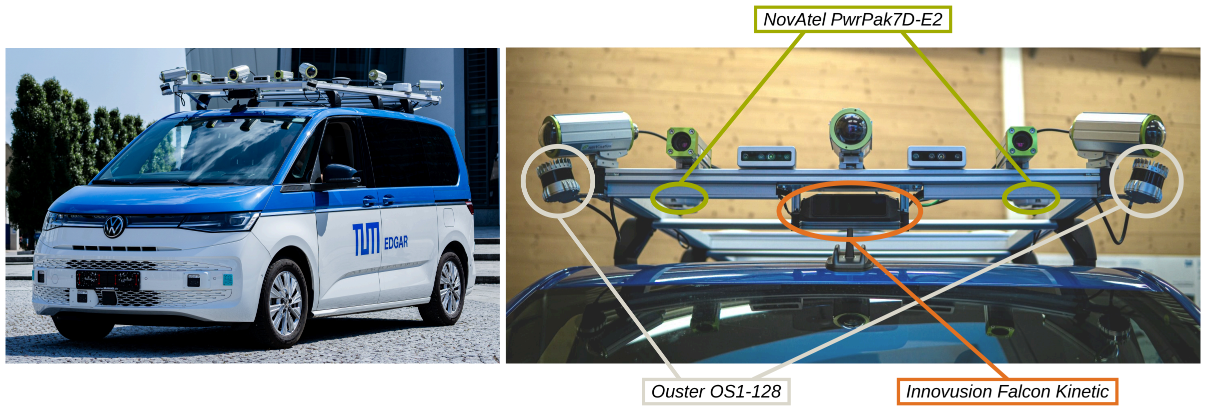
Therefore, a novel mapping pipeline for the fully equipped research vehicle EDGAR is developed and evaluated. Based on the previous Chapter 4, this chapter will further transfer the findings from localization and mapping on race tracks to public roads. This chapter aims to build an applicable and arbitrarily expandable localization and mapping pipeline for the research community in general and especially the research vehicle EDGAR. The main focus is on LiDAR mapping for localization and semantic mapping to enable motion planning, prediction, etc. Therefore, different existing algorithms and data sources are combined to create the first open mapping pipeline for HD maps for automotive applications. The map format is based on open standards to fulfill the requirements to be compatible with the *Autoware* based software stack. A preliminary version of the pipeline presented was published in [18]. This chapter builds on this work and adds additional functionalities, areas, evaluation, and results.

5.1 EDGAR Research Vehicle

Before the developed pipeline is presented, the autonomous research vehicle EDGAR is introduced in more detail in this section. As the base vehicle platform, the EDGAR research vehicle uses a *Volkswagen T7 Multivan e-Hybrid*. This is equipped with various sensors, computers, actuators, etc. Figure 5.1 (a) shows the vehicle in front of the TUM main building in Garching. A more detailed photo of the sensor rack, viewed from the front of the vehicle, is depicted in Figure 5.1 (b). The most relevant sensors for the remainder of this work are highlighted, namely GNSS and LiDARs.

Eight cameras ensure $360°$ vision around the vehicle, and six RaDARs to the front and rear can detect vehicles at a large distance and measure their velocity by exploiting the *Doppler* effect. An RTK-corrected GNSS receiver from *NovAtel* provides global positioning data. Under open sky and good conditions, it reaches cm-level accuracy and can be used as a ground truth for localization. In addition, it includes an IMU that allows for enhanced localization and state estimation through INS. The receiver outputs a filtered INS solution calculated on the internal processor and the raw sensor data. Thus, GNSS and IMU data can also be fused on the main computing platform.

In addition to the GNSS receiver, the LiDARs are the primary sensors for localization and mapping. EDGAR comprises a total of four laser scanners to cover the area around the vehicle and high ranges to the front and rear. Two spinning *Ouster OS1-128* $360°$ LiDARs with a vertical FOV of $45°$ are placed on the front corners of the vehicle. Especially for long-range perception to the front and rear, two *Innovusion Falcon Kinetic* are mounted at the front and rear centers of EDGAR. They provide a horizontal FOV of $120°$ and a vertical



(a) EDGAR research vehicle on the TUM campus. (b) Roof-mounted sensor rack.

Figure 5.1: EDGAR research vehicle and detail photo of the roof-mounted sensor rack.

FOV of $25\,°$. At $10\,\%$ reflectivity, they have a detection range of $250\,\mathrm{m}$. Due to their solid-state design, the FOV and a specific Region Of Interest (ROI) can be changed at runtime depending on the current situation. The LiDAR setup of EDGAR with the FOVs of the sensors mentioned is depicted in BEV in Figure 5.2. The multi-LiDAR configuration induces inherent challenges, mainly spatial calibration, temporal synchronization, and synchronized triggering. The following sections will elaborate on these challenges and the proposed solution approaches.

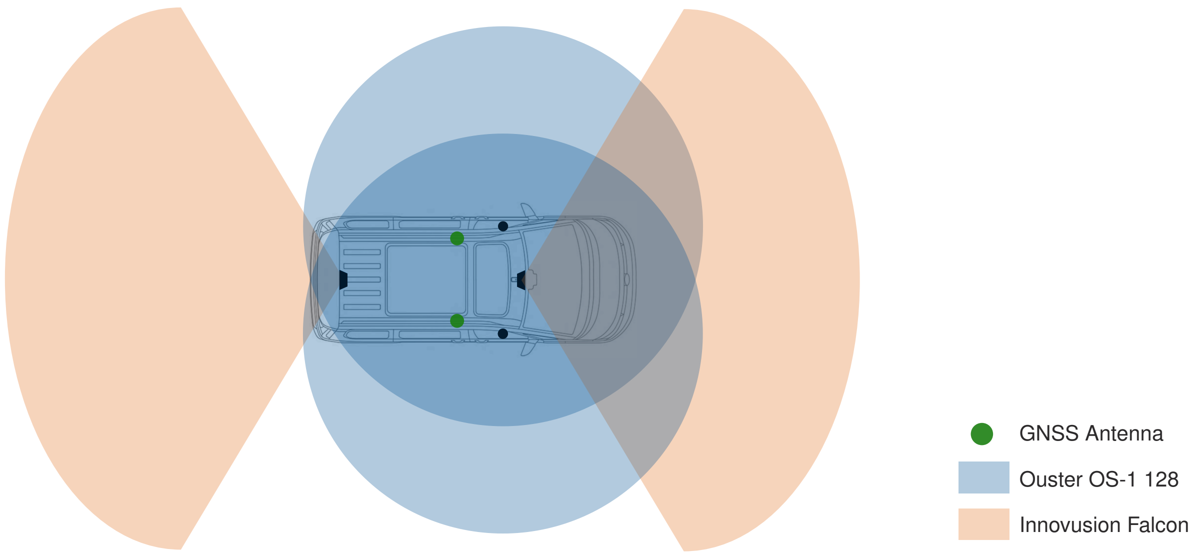


Figure 5.2: LiDAR setup of the research vehicle EDGAR.

An overview of the whole perception sensor setup and the compute configuration is given in Table 5.1. The vehicle serial sensors can be accessed via a Controller Area Network (CAN) interface and are connected to the main computer. A grandmaster clock is installed to provide a time signal for precise PTP synchronization for all sensors.

Figure 5.3 shows the concatenation of four individual LiDAR point clouds from EDGAR transformed into a common frame. The reflections of each LiDAR are painted in the corresponding color for better distinguishability.

Table 5.1: Overview of the perception hardware components of EDGAR. The FOV is given horizontally and vertically (h x v).

Component	Manufacturer	Model	Frame rate	FOV (h x v)
GNSS Receiver	*NovAtel*	*PwrPak7D-E2*	20 Hz GNSS, 100 Hz IMU	-
2x LiDAR Mid-Range	*Ouster*	*OS1-128*	10 Hz - 20 Hz	360° x 45°
2x LiDAR Long-Range	*Innovusion*	*Falcon Kinetic*	10 Hz	120° x 25°
8x Cameras	*Basler*	*acA1920-50gc*	up to 40 Hz	depends on lens
6x RaDAR	*Continental*	*ARS430*	10 Hz	±60° x 2.2° (short range), ±9° x 2.2° (long range)
Compute Platform	*InoNet*	*Mayflower B17*	-	-
Network Switch	*Netgear*	*M4250-40G8XF-PoE+*	-	-

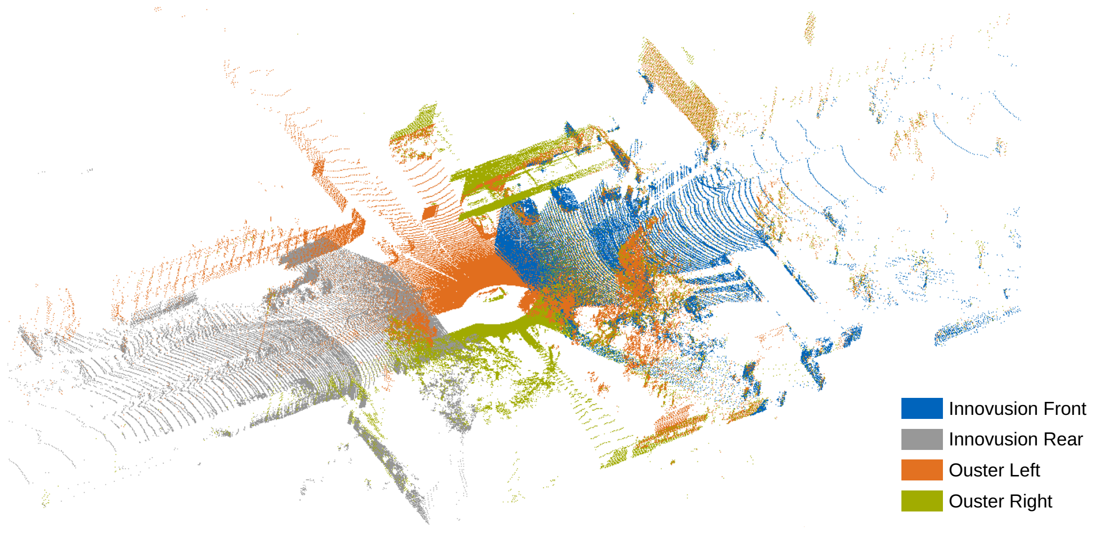


Figure 5.3: Concatenated point cloud from all four LiDARs of EDGAR.

5.2 Pipeline Overview

This chapter introduces the general pipeline of the EDGAR software stack and, specifically, the mapping and localization pipeline.

Generally, the software stack builds on the *Autoware* open-source framework [341]. The stack follows the standard AD approach as presented in Section 1.2 and is based on the *ROS 2* middleware. It provides baseline implementations for all common software modules. *Autoware* is divided into two module collections, namely *Core* and *Universe*. *Autoware.Core* includes the most important modules needed for basic autonomous driving. It adheres to certain code standards and is entirely written in *C++*. *Autoware.Universe* provides additional packages to expand the software's functionality and thus the usable ODDs. It simplifies community contributions by reducing code contributions' specifications and requirements.

The overall concept of the localization and mapping pipeline developed for EDGAR includes two separate strands: offline mapping and online localization, with both sharing common algorithms. The concept overview is depicted in Figure 5.4.

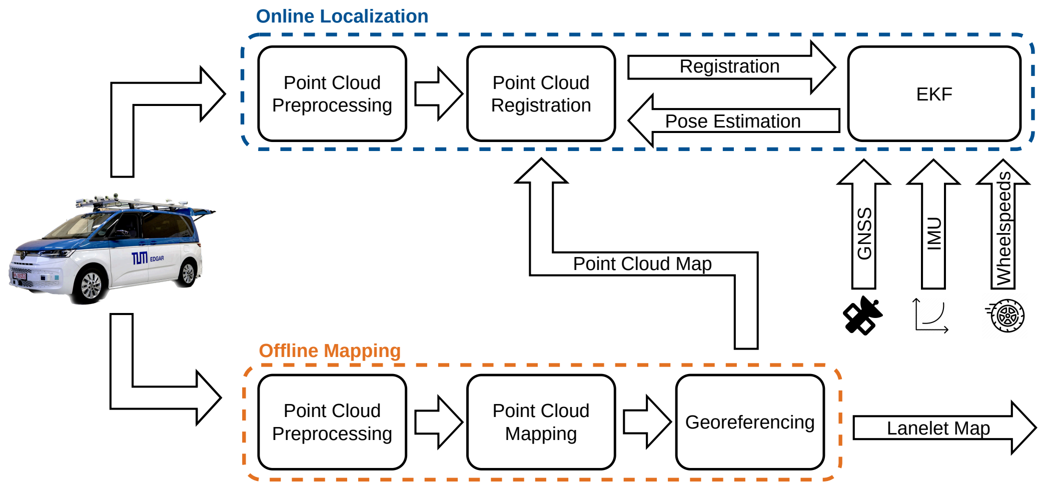


Figure 5.4: Overview of the localization and mapping pipeline for the EDGAR research vehicle.

In the offline mapping pipeline, there are two outputs to be generated:

- A precise 3D point cloud map for online LiDAR localization
- A vector map of the lanes in the *Lanelet2* format for motion planning, etc.

The point cloud mapping pipeline is built on the *KISS-ICP* [82]. This algorithm showed the best results in terms of applicability and generalization. Different approaches are introduced for vectorized lane mapping based on online data and vehicle sensors. The finally chosen approach leverages crowd-sourced OSM data for semantic map generation. In addition, the *FlexMap Fusion* tool, developed within the scope of this thesis, is introduced. It allows for georeferencing and fusion of different data sources. The mapping pipeline will be presented in more detail in Section 5.4.

The online localization module also uses the adapted *KISS-ICP* and is similar to the point cloud mapping pipeline but uses a pre-built point cloud map. An EKF uses the information from the LiDAR localization and fuses it with additional vehicle sensors (GNSS, IMU, and wheel odometry) to output a high-frequency ego state estimate.

5.3 Multi-LiDAR Scan Registration

Subsection 4.1.4 reveiled several challenges for multi-LiDAR scan matching. Therefore, the LiDAR fusion scheme proposed in this section is implemented for the point cloud registration of EDGAR. Instead of fusing all LiDARs as shown in Figure 5.3 and processing it afterward, each LiDAR is processed on its own. Figure 5.5 depicts the LiDAR fusion scheme integrated into *KISS-ICP*. *KISS-ICP* offers a LiDAR odometry and focuses on robustness and real-time capability. First experiments showed that it outperforms other algorithms with custom data, even those that showed superior performance on benchmark datasets. After an optional preprocessing, e.g., voxelization, geometric filtering, etc., of the single point clouds, they are all transformed into a common base frame. Here, the projection of the center of the rear axle to the ground plane is chosen as the base frame. Therefore, external spatial calibration data that have been identified offline before are used. In this approach, every point cloud can be registered with its own timestamp, and no assumptions or compensations are needed for point cloud fusion and registration. Synchronization errors can be corrected through scan matching without inducing map inconsistencies. After each registration, a motion update is performed. In this use case, a simple forward projection of the last registration result was performed. However, in future work, these pose estimates could be treated as separate motion updates in a Kalman

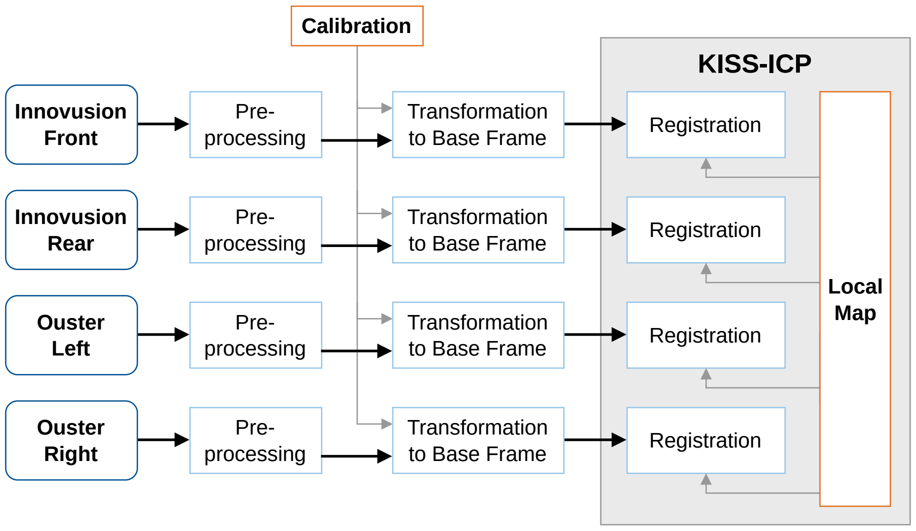


Figure 5.5: LiDAR fusion scheme for a robust point cloud registration for EDGAR's multi-LiDAR setup.

Filter to also account for unequal precisions. The presented scheme for LiDAR fusion and scan matching is used both for point cloud map building and for localization in existing pre-built maps. Therefore, two modes for mapping and localization are implemented with the same underlying scan-matching methodology. In the mapping mode, the local map is created at runtime and used for registration. In localization mode, a 3D point cloud map is loaded, and the LiDAR frames are registered online. For initialization, the initial pose has to be input. If available, this can be done using GNSS. Otherwise, the initial pose has to be set manually.

In summary, the following changes have been made to *KISS-ICP* [82, 342]:

- Implementation of multiple point cloud subscribers so that the node can listen to every LiDAR.
- Transformation of every point cloud to the base frame (rear axle center ground).
- Global map loading for localization on existing point cloud maps.
- Dynamic pose initialization for localization mode in an existing map coordinate frame.
- Motion update for each registration.

5.4 HD Mapping Pipeline

This section presents the developed open-source mapping pipeline focusing on quick mapping of unmapped areas. The LiDAR registration methodology from Section 5.3 is used for the point cloud mapping process. The last part presents the semantic mapping pipeline, e.g., the creation of lane-level maps in the *Lanelet2* format.

5.4.1 Map Format

To provide compatibility with the *Autoware* framework and the EDGAR research vehicle software environment, the map has to fulfill the following requirements:

1. 3D point cloud map as *.pcd* file and semantic lane map in the *Lanelet2* format for compatibility with *Autoware*.

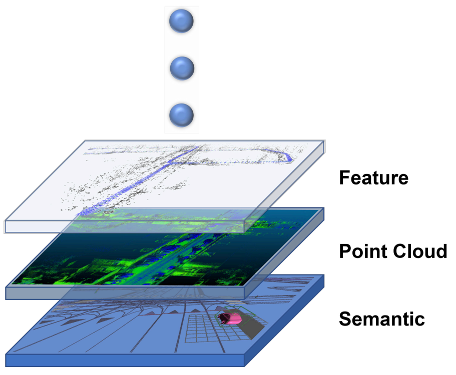


Figure 5.6: Expandable multi-layer HD map structure.

2. Arbitrary expandability for integrating, evaluating, and testing novel localization algorithms for research purposes.

The resulting HD map structure aligns with the state of the art as presented in Subsection 2.3.1. Figure 5.6 illustrates the structure with the stacked layers of the map. The different layers are used for different use cases, such as localization, motion planning, motion prediction, etc. All layers share the same coordinate system so that positions can be used across different layers without any transformations. For the first implementation and integration with the *Autoware* framework, the two base layers, namely a semantic layer in the *Lanelet2* format and a 3D point cloud map, are needed. Thus, the following will focus on these layers and their generation. However, for future research, for example, feature maps could be used to evaluate visual mapping and localization algorithms within the whole EDGAR software environment, thus leading to more general results than evaluating algorithms isolatedly.

5.4.2 Point Cloud Mapping

The LiDAR scan registration for the point cloud mapping is based on the algorithm presented in Section 5.3. However, since *KISS-ICP* is a LiDAR odometry only and does not offer loop closure capability, global optimization, or GNSS integration, post-processing steps are required to generate globally consistent maps [82]. Therefore, the *Interactive SLAM* toolbox [317] is used. The estimated ego poses by the *KISS-ICP* and their corresponding point clouds are imported for manual and semi-automatic refinements. Additional constraints, for example, loop closure, can be set to make the map globally consistent. Also, errors and wrong constraints can be manually removed to improve the point cloud map quality. After this, an automatic fine registration is applied by running scan registration with the newly set constraints. The final map is built by transforming all point clouds to their corresponding positions and accumulating them. It can be saved as a 3D point cloud in the *.pcd* format to be used in the online localization mode. To control and limit the size of the final point cloud map, only keyframes are exported. The requirements for keyframes (e.g., change in distance and rotation) can be adjusted to find a good trade-off between file size and map resolution. After this, voxel filtering with a definable voxel size is also applied to reduce the amount of data further.

5.4.3 Semantic Mapping

Compared to point cloud mapping, semantic mapping of lanes and road geometries poses different challenges. For comprehensive semantic road mapping, road markings and boundaries, such as walkways, road ditches, etc., must be detected and mapped. Road markings can exhibit color, texture, and curvature variations, often making them difficult to distinguish from surrounding objects. For road boundaries, generating heuristics that generalize well across different ODDs is difficult. Deep learning-based methods can be used; however, as the related work revealed, they show high data dependency and thus need labeled training data from a specific use case.

Additionally, semantic mapping requires an understanding not only of the geometric texture of the environment but also of the contextual relationships. For example, lane markings and other road elements, such as lane boundaries, intersections, and traffic signs, are related and must be interpreted correctly. These factors introduce an additional layer of complexity, which demands more sophisticated machine learning techniques to accurately interpret and represent lane information.

As Subsection 2.3.3 revealed, this complexity leads to a lack of performance of conventional algorithms for the reliable detection and mapping of lane information. The same was determined for the EDGAR project [343]. Both conventional and learning-based approaches failed to generalize well enough to allow for a applicable, generalizing mapping pipeline. Therefore, current research focuses on the direction of deep learning-based online HD mapping as presented in Subsubsection 2.3.3. However, these approaches are highly data-dependent with a limited generalization to other domains, e.g., other sensor setups, locations, seasons, etc. As sufficient data, especially labeled data, from EDGAR are not available, such approaches are not applicable at the current state and will be further researched in future work.

To still be able to quickly create maps of previously unmapped areas without having to manually select the lanes, OSM was chosen as the main data source [344]. It provides the geometric and semantic information of the lanes, such as street type, speed limits, etc. The following will describe the approach to process this information and generate maps in the *Lanelet2* format.

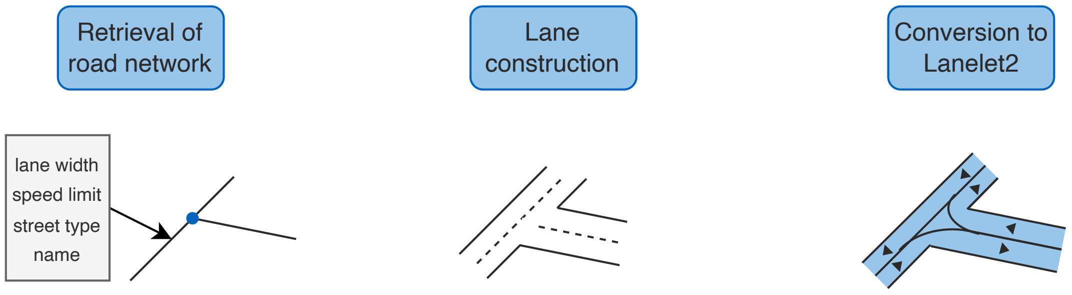


Figure 5.7: Generation of the lanelet from OSM data. [344]

The high-level process of generating the lanelet map from OSM data is depicted in Figure 5.7. As OSM contains only the center lines of the street and the number of lanes, the actual lane geometry has to be estimated. Therefore, the width of each lane - if not included in the data - must be estimated based on the type of road, which the OSM data contain as the corresponding `highway` tag. Based on the German regulations of the *Research Association for Road and Transportation (Forschungsgesellschaft für Straßen- und Verkehrswesen e.V.)* [345], the standard total road width (German: *Regelquerschnitt, RQ*) specifies the setup of the entire road and the individual lane widths. The extracted values used for the generation of lanelets are given in Table 5.2. To generate the final lanelet in the *Lanelet2* format, additional information, such as driving direction, speed limit, etc., is added to the corresponding lanes. If refinement of the semantic map is needed, e.g., in intersections, the tool *VectorMapBuilder* [346] can be used.

In this step, not only the road geometry is extracted, but also additional semantic information contained in the OSM data. This can be speed limits, pedestrian crossings, traffic lights, etc.

Table 5.2: Assumed lane width based on the OSM `highway` tag [344, 345].

Description	Assigned OSM types	Standard total road width	Lane width
Highway	motorway	RQ 29.5	3.75 m
Large main road	trunk	RQ 10.5	3.5 m
Medium main road	primary, secondary, busway	RQ 9.5	3 m
Inner city and access roads	tertiary, unclassified, residential, service, living_street	RQ 7.5	2.75 m

5.4.4 Georeferencing

For several reasons, georeferencing is an important step in creating precise HD maps. It allows the use of GNSS data for localization in the map and fusion with LiDAR-based localization approaches. Without a georeferenced map, no fixed transformation exists between the GNSS and the point cloud map coordinate systems. Moreover, a joint georeferencing of the point cloud and the *Lanelet2* maps guarantees their alignment. As described above, the point cloud maps are generated and optimized without using GNSS data. For the extraction of semantic maps, georeferenced data from OSM is used. However, this also has to be post-processed to guarantee the consistency of all map layers. Therefore, a georeferencing and map data fusion tool called *FlexMap Fusion* is developed. It is available as open source software under https://github.com/TUMFTM/FlexMap_Fusion. The structure of the tool is depicted in Figure 5.8.

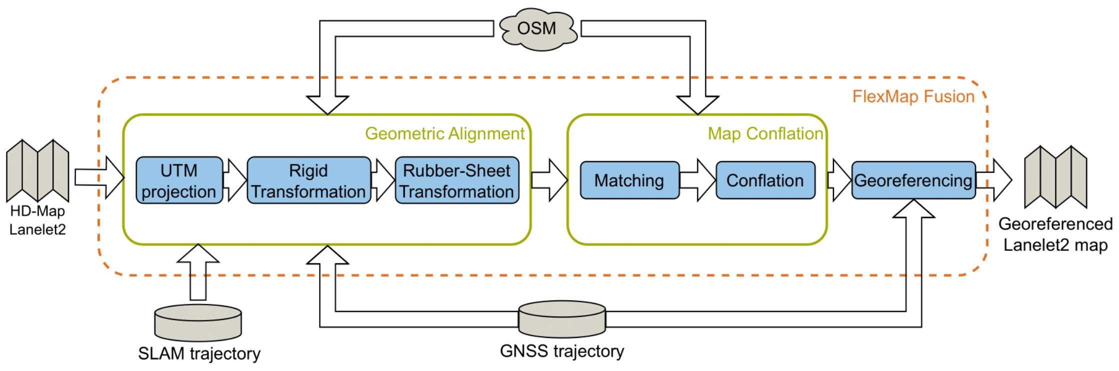


Figure 5.8: Structure of the *FlexMap Fusion* conflation tool [19].

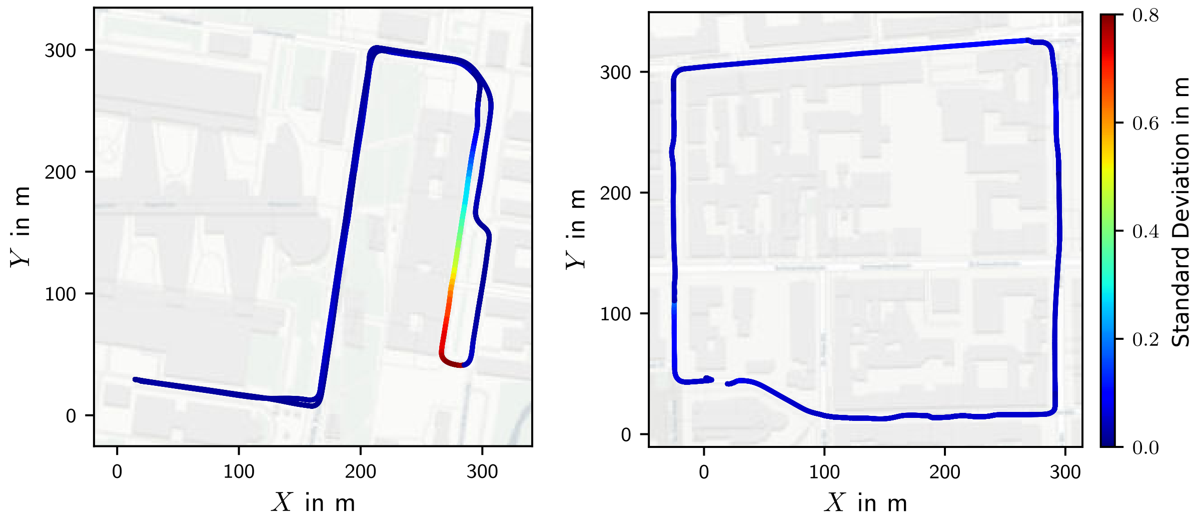
To use local Cartesian coordinates, a map origin point is defined. It is either the starting point of the driven trajectory, or it can be manually set. For the projection from geographic GNSS coordinates to Cartesian coordinates, the Universal Transverse Mercator (UTM) transformation is used. The process of fusing two maps that represent the same environment differently (here: point cloud and lanelet map) is usually referred to as conflation. *FlexMap Fusion* comprises three modules for map conflation and georeferencing: During **Map Alignment**, the maps are aligned with the RTK corrected ego GNSS trajectory in a projected local coordinate frame. First, a rigid *Umeyama* transformation [347] is used. It rigidly translates and rotates one map to find the optimal alignment without changing the maps' geometric appearance or trajectories. To actually align the maps and remove the accumulated error from map creation, e.g., in SLAM, a piecewise linear rubber-sheet transformation is applied [348]. Therefore, the area to be mapped is triangulated using control points. These control points can either be set manually (e.g., visually corresponding points, such as sharp turns) or be taken from the GNSS trajectory (only points with low estimated standard deviation are used). The algorithm applies linear transformations in each triangle based on the deviation of the maps at the control points. This alignment enables the conflation of the different map data in the **Map Conflation** module. This step uses a buffer-growing algorithm to identify the corresponding road sections between the maps. After this, the information can be transferred between different maps. For example, the speed limit from OSM can be added to a lanelet map built using sensor data from the ego vehicle. The projection of the aligned, conflated, and consistent maps to global coordinates in the **Map Georeferencing** step generates the final output. This allows the map to be used for localization using GNSS data directly. More details on the exact implementation of the algorithms can be found in [349].

5.5 Results and Discussion

This section presents the results and performance of the presented mapping pipeline [344]. Therefore, two routes, one at the TUM campus in Garching and one in the Munich city center are mapped, and the results are evaluated. Extending the results, the discussion will identify open points and future research directions.

Experimental Setup

Two custom datasets that have been recorded with EDGAR are used. One was recorded at the TUM Campus Garching (*Garching* dataset) and the other one in the center of Munich between the main station and the Theresienwiese, referred to as the "*Wiesn*" dataset. Figure 5.9 shows the driven GNSS trajectories in BEV. The standard deviation estimated by the GNSS receiver internally is represented by the color at each corresponding position. In the *Garching* dataset, the area behind the *Galileo* building exhibits the highest uncertainty, with the standard deviation reaching a maximum of $0.814\,\mathrm{m}$. The mean standard deviation values $0.089\,\mathrm{m}$. Figure 5.9 (b) depicts the route of the *Wiesn* dataset. The GNSS signal demonstrates significantly reduced standard deviations, with the maximum being $0.184\,\mathrm{m}$. The mean standard deviation values $0.053\,\mathrm{m}$. As estimated standard deviations from the *NovAtel* GNSS receiver are in the cm range for the largest part of the trajectories, GNSS can be used as a reference signal for the ego position to evaluate the map quality.



(a) *Garching* dataset. Behind the Galileo building, the GNSS signal shows the highest uncertainty.

(b) *Wiesn* dataset. It shows comparably less uncertainty throughout the trajectory.

Figure 5.9: Recorded GNSS trajectories of the two evaluated datasets. The color represents the standard deviation as estimated by the *NovAtel* receiver.

The progress of the parallel setup and development of the hardware platform and sensor driver implementation can be seen in the contained data. As the *Garching* dataset was recorded a few weeks earlier, it only comprises the two *Ouster OS1-128* LiDARs and no time synchronization. In the *Wiesn* dataset, all four

LiDARs are available, and they are synchronized using the PTP standard. By this, the exact timesteps and the time between the recording of the LiDAR point clouds are known. Also, the external spatial calibration between the LiDARs was improved. However, due to the increased data payload, in the *Wiesn* dataset, the two *Ouster* LiDARs showed a decreased frame rate of only around $3\,\mathrm{Hz}$. Also, the images from the front cameras are available in the dataset. These will be used for a better understanding of the corresponding situations in the discussion section. Table 5.3 comprises the data contents of the two evaluation datasets.

Table 5.3: Overview of the two EDGAR datasets for evaluation.

	Garching	***Wiesn***
Start of Recording	2023-07-20, 16:41	2023-09-29, 14:40
Scenery	semi-urban	urban
Start Location	$48.2644°$ N, $11.6690°$ E	$48.1367°$ N, $11.5519°$ E
Sensor Setup	RTK-GNSS, 2x LiDAR	RTK-GNSS, 4x LiDAR, PTP
Driven Distance	$1565.3\,\mathrm{m}$	$1189.3\,\mathrm{m}$
Duration	$310.4\,\mathrm{s}$	$766.2\,\mathrm{s}$

The following section will demonstrate the working principle and the results of the described mapping pipeline on these two datasets. Therefore, first, the estimated ego trajectory is quantitatively evaluated after each mapping step:

- LiDAR point cloud mapping using the customized *KISS-ICP*
- Loop closure and global optimization with *Interactive* SLAM
- Georeferencing with manual control points using the *FlexMap Fusion* tool

Following, the OSM-based semantic *Lanelet2* maps are evaluated qualitatively. As no ground truth is available, only a visual evaluation can be carried out.

Results

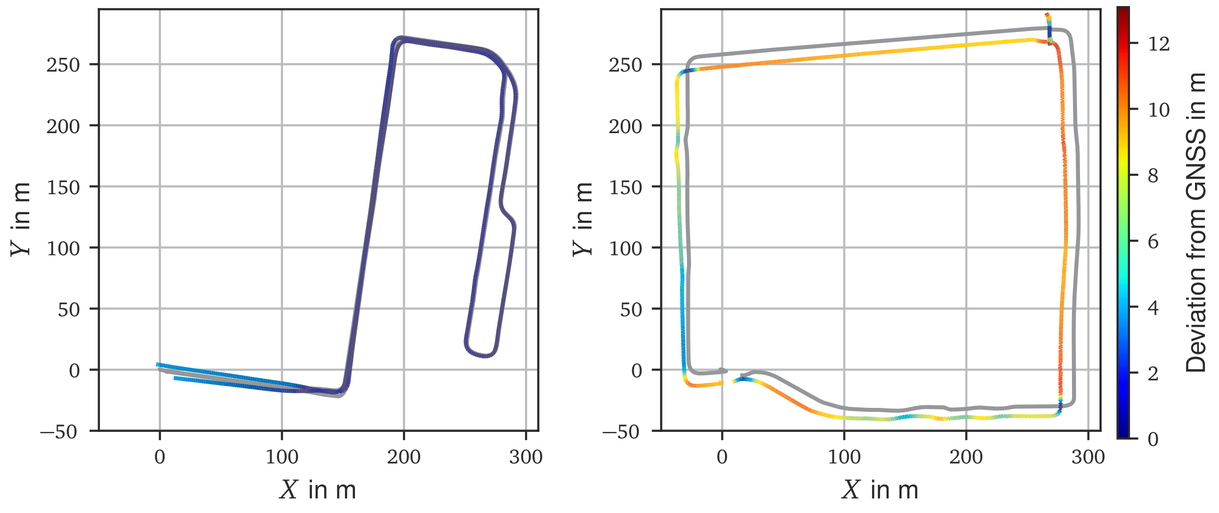
The results after each mapping step are summarized in Table 5.4. The following section will relate to this table and explain the single steps of the pipeline. As elaborated in Subsection 2.3.2, it is assumed that the required map precision is in a range of $10\,\mathrm{cm}$ to $20\,\mathrm{cm}$. The created maps show a final median precision of $35.1\,\mathrm{cm}$ and $18.8\,\mathrm{cm}$. This means that the maps are close to the desired range. However, the maximum errors will have to be minimized in future work.

Table 5.4: Decrease of the mapping trajectory error after each mapping step. The RTK-GNSS signal filtered for outliers is used as a reference trajectory. All values are given in m.

	Garching					***Wiesn***				
	RMSE	Mean	Median	Std. Dev. σ	Max	RMSE	Mean	Median	Std. Dev. σ	Max
KISS-ICP	1.488	1.109	0.844	0.992	5.455	8.737	8.465	9.096	2.163	13.085
Interactive SLAM	0.706	0.550	0.488	0.443	2.321	0.775	0.630	0.620	0.451	1.928
Georeferencing	0.549	0.425	0.351	0.348	1.907	0.278	0.217	0.188	0.173	0.929

The first step of the map generation process is to create a first point cloud map with the adapted *KISS-ICP* LiDAR odometry. For the *Garching* dataset, only the two *Ouster* LiDARs could be used; the *Wiesn* dataset makes use of all four laser scanners. As no loop closure or GNSS was used in the first LiDAR mapping step, the trajectories show an increasing drift over time. Since the *KISS-ICP* does not have a global reference, the

Umeyama algorithm [347] is used to align the SLAM trajectory with the GNSS trajectory and calculate the error. The aligned trajectories are shown in Figure 5.10.



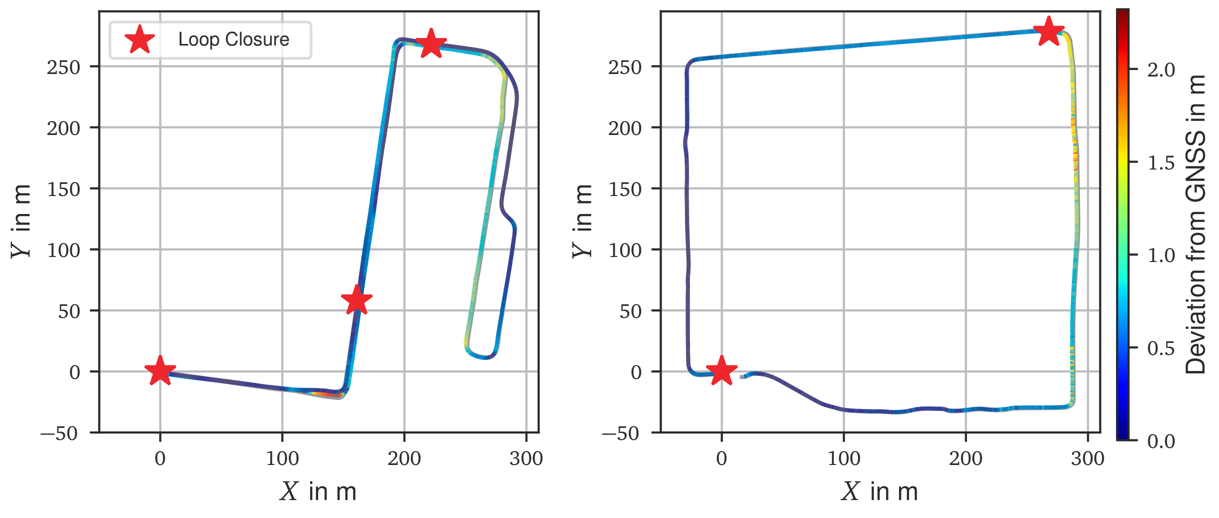
(a) *Garching* dataset. (b) *Wiesn* dataset.

Figure 5.10: Deviation between the GNSS and the aligned *KISS-ICP*. The deviation is calculated to the spatially closest point.

It becomes evident that the *Garching* dataset shows significantly lower deviations. However, it should be mentioned that the most significant part of the error in the *Wiesn* dataset comes from the outlier in the top right corner. This leads to a shift of the entire trajectory in the rigid alignment process. Another problem can also be seen with this error assessment. As no temporal relations are used, only the error to the spatially closest point can be calculated. By this, longitudinal errors can be neglected, for example. This behavior can be observed when the trajectories cross, e.g., in the top left corner of the *Wiesn* dataset. The error is assumed to be zero, even if there is still a deviation between the estimated trajectory and the reference.

As the *KISS-ICP* does not have GNSS integration or loop closure capability [82], the *Interactive* SLAM [317] is used to manually add loop closure and additional constraints to the *KISS-ICP* trajectory. The *Interactive* SLAM then globally reoptimizes the entire trajectory and thus compensates for the accumulated drift. The manually added loop closure constraints for both datasets are shown in Figure 5.11, marked as red stars. In the *Wiesn* dataset, additionally, the faulty registered point clouds in the top right corner and their corresponding edges have been removed. The improved results after application of the *Interactive* SLAM can be found in Table 5.4. As the *Wiesn* dataset only provides one big loop, the error increases to its maximum at the top right. This is the position furthest away from the loop closure at the beginning and end of the trajectory. In the *Garching* dataset, the same road is driven in two directions, and thus, more loop closure constraints can be added, leading to a smaller error in the trajectory.

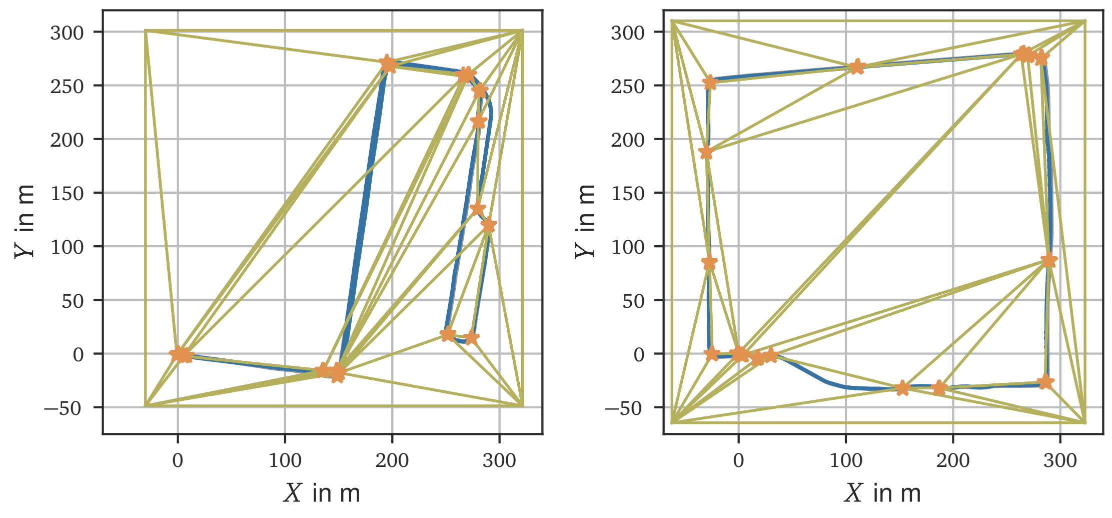
The final step of the point cloud mapping pipeline marks the georeferencing using the custom-developed rubber-sheeting and georeferencing tool *FlexMap Fusion* [19]. For georeferencing, distinct point pairs between the GNSS trajectory and the trajectory after *KISS-ICP* and *Interactive* SLAM are selected. These are marked as orange stars in Figure 5.12. The area around the trajectory is triangulated according to these control points. As the error is known for each control point, it can be interpolated for each triangle, leading to a linear transformation that is steady on the edges. The selection of suitable control points is especially difficult on long straights. This results in the straights being relatively far away from the correction points and thus showing increased deviations. This behavior can be seen in Figure 5.13.



(a) *Garching* dataset.

(b) *Wiesn* dataset.

Figure 5.11: Deviations between the GNSS and the aligned *Interactive* SLAM trajectory. The red stars show the positions of manually added constraints, such as loop closures.



(a) *Garching* dataset. Nineteen control points are selected.

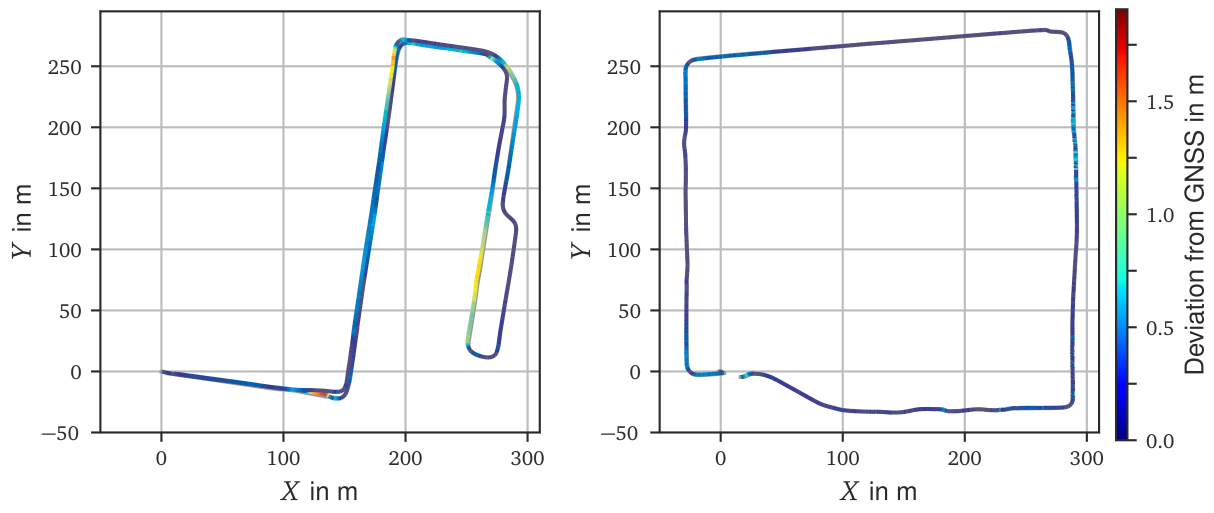
(b) *Wiesn* dataset. Twenty control points are selected.

Figure 5.12: Triangulation for the rubber-sheet transformation between the GNSS and the *Interactive* SLAM trajectory. The orange stars illustrate the control points.

The final errors of the mapping pipeline are included in Table 5.4. The *Wiesn* dataset shows significantly smaller deviations from the GNSS signal, which is used as a reference trajectory. A combination of several factors can explain this:

- Two additional LiDAR sensors for long range and improved coverage to the front and the rear of the vehicle.
- Added PTP synchronization for known time differences between all point clouds.
- Geometrically different environment: campus and urban.

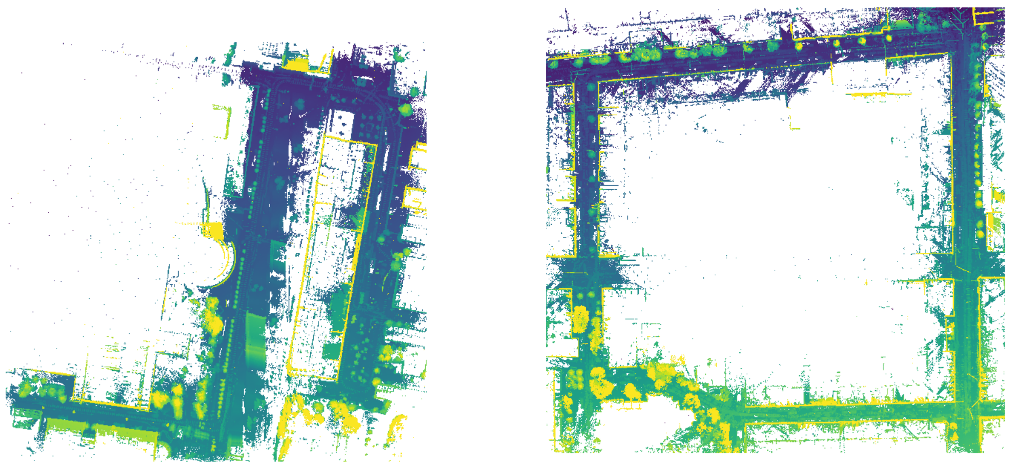
After calculating and correcting the trajectory positions based on their corresponding point clouds, all single scans are accumulated to obtain dense 3D maps. These can be used for online map-based localization. Figure 5.14 shows the generated point cloud maps for a qualitative impression. A closer look at the maps



(a) *Garching* dataset. (b) *Wiesn* dataset.

Figure 5.13: Deviations between the GNSS and the rubber-sheeted SLAM trajectory.

reveals a higher density for the *Wiesn* dataset. This is because scans from four LiDARs are accumulated instead of only the two rotating *Ousters*.

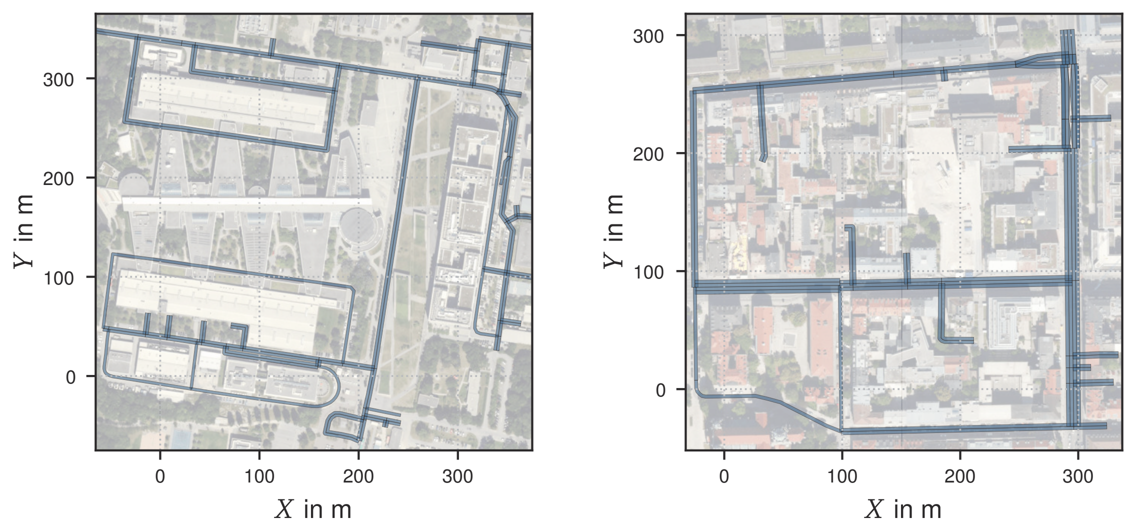


(a) *Garching* dataset. (b) *Wiesn* dataset.

Figure 5.14: Created point cloud maps for the *Garching* and *Wiesn* datasets. The color encodes the z-coordinate.

The final part of this section elaborates on the results of the semantic map generation. Therefore, the road information of OSM is extracted for the corresponding areas. Following the process presented in Subsection 5.4.3, maps in the *Lanelet2* format can be created. As there is no ground truth for the lane geometries, this section will conduct a qualitative and visual evaluation of the results. Therefore, the created semantic maps are overlaid with orthographic and georeferenced photos provided by the *OpenData* dataset of the Bavarian government [350]. In addition, the lanes are overlayed with the previously created point cloud maps to check for consistency across the different map layers. Figure 5.15 shows the generated *Lanelet2* maps and the corresponding aerial orthophotos.

More detailed images from both the *Garching* dataset as well as the *Wiesn* dataset are shown in Figure 5.16. The figures disclose that the lanes align well with the orthophotos. Even if a quantitative evaluation cannot be carried out, the OSM based lanelet map builds a sufficient initial base for autonomous driving in urban

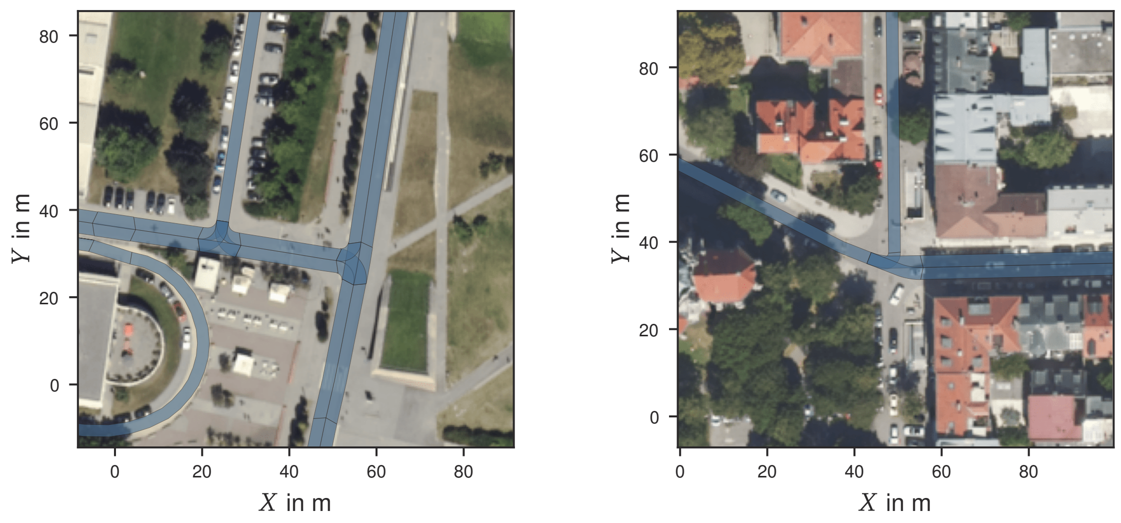


(a) *Garching* dataset.

(b) *Wiesn* dataset.

Figure 5.15: Result of OSM-based vector map generation. Orthophotos provided by *OpenData* [350].

regions to follow the lanes. However, it cannot be ruled out that after the first tests, the map must be refined manually or based on onboard sensor data. The *FlexMap Fusion* tool offers the possibility to adjust a lanelet map based on OSM data to an existing point cloud map in case of inconsistencies.



(a) *Garching* dataset.

(b) *Wiesn* dataset.

Figure 5.16: Plot of lanes (blue) on top of orthophotos provided by *OpenData* [350].

For the final result of the HD mapping pipeline, the point cloud maps are combined with the *Lanelet2* maps. As the two map layers are generated from different data sources, validating their correspondence, which the georeferencing should give, is essential. Figure 5.17 shows the two generated maps with their overlaid layers in the BEV. On this basis, the point cloud map can be used for localization, and the semantic layer can be used for path planning, motion prediction, etc. The presented HD map is fully compatible with the *Autoware* software stack.

For a more detailed check of the alignment of the maps, Figure 5.18 shows zoomed-in sections of the two maps. The generation processes for the point cloud map and the lanelet map involve uncertainties in the data

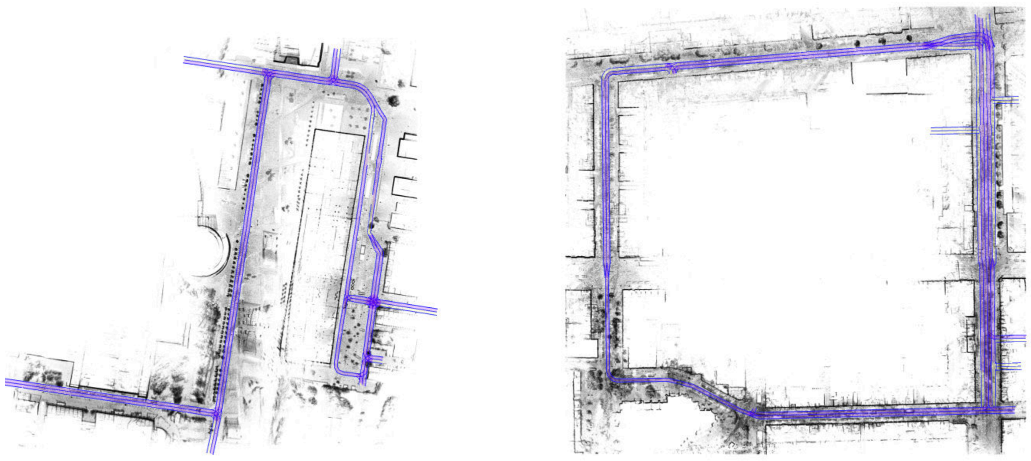

(a) *Garching* dataset. (b) *Wiesn* dataset.

Figure 5.17: Overlay of the point cloud map and the *Lanelet2* map.

themselves, possible errors in data processing, heuristics for lane mapping, and georeferencing measures. Despite these factors, the derived map layers show accurate alignment. The derived lanes represent drivable spaces and do not intersect with static obstacles for almost the entire mapped area. However, Figure 5.18 (b) reveals that on the right side, a parked car intersects with the map lane. It is not clearly visible if this represents an error in the generated map or if the car is actually parked partially on the road. Such occurrences will have to be evaluated in more detail. Therefore, a validation and evaluation pipeline should be developed to quantitatively assess the results. Also, on the left side of this section, it can be seen that there is a slight deviation between the lanelet map and the point cloud. This slight drift probably stems from the point cloud mapping process, which relies on manual control points. Thus, uncertainties and map errors can occur, especially between these points.

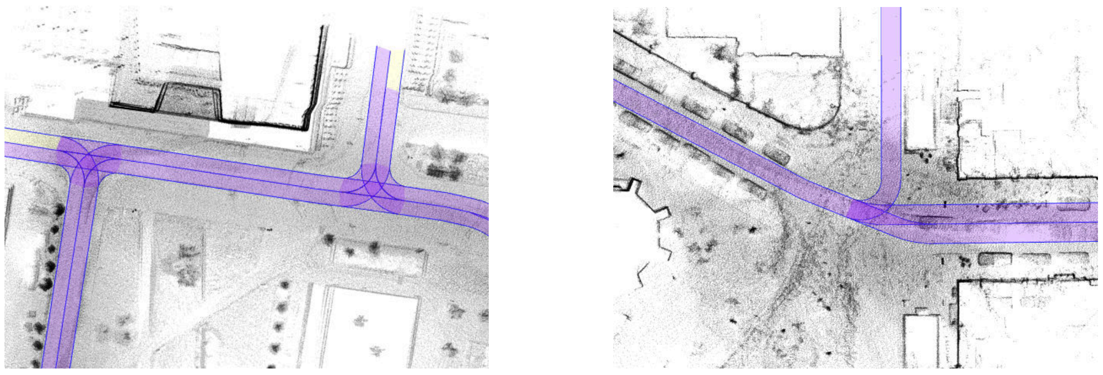

(a) *Garching* dataset. (b) *Wiesn* dataset.

Figure 5.18: Overlay and alignment of the generated point cloud and vector map in the *Lanelet2* format.

In summary, Table 5.4 shows that after the whole mapping pipeline, accuracies can be reached that allow online use of the created maps. An additional visual investigation proved the alignment of the point cloud maps and the lanelet maps. However, the maximum errors of the maps are still in the range of meters. Therefore, it is advised that the HD maps are validated in shadow mode before deploying them to the autonomous driving software stack and entirely relying on them. Also, online monitoring of the HD maps and detection of map errors should be implemented. An important point for future research will be the evaluation and assessment of maps to guarantee correctness and accuracy.

Discussion

Based on the results presented, the whole pipeline will be discussed below. First, some topics of the online localization pipeline will be considered. The remainder of this section will focus on discussing the HD mapping pipeline and defining points for future research.

Localization:

First experiments and tests proved that the created HD maps can be used for online localization both with the standard *Autoware* localization and the custom *KISS-ICP* localization module. However, some points are still open and will be discussed first. In the current implementation, pose initialization is done by taking a pose from the GNSS receiver. This can cause problems in case of occlusion or GNSS disturbances such as multipath reflections. The current backup is a manual pose initialization, which is also not feasible in every case. The topic of pose initialization is also part of recent research, and some approaches are already available that will have to be implemented and tested with EDGAR [351].

Another possible improvement of the localization module is estimating a dynamic covariance matrix. In the way it is implemented, and according to the state of research, a fixed covariance is assigned to the localization output. Usually, the values are based on observations and experience. However, the uncertainty of the localization module can vary greatly depending on the current vehicle state, sensor performance, the appearance of the environment, etc. Normal ICP based algorithms can only output information about the point-wise distance. This does not have to correlate with the actual accuracy of the scan registration, as already shown in Subsubsection 4.1.2. There are still possibilities to estimate a dynamic covariance. This could be done by deriving heuristics based on past experiences, e.g., runtime variation or point distribution, or by extracting uncertainty information directly from the registration algorithm.

Finally, more evaluation is needed to obtain more information on the performance of the localization module. For the scope of this work, the amount of available data and scenery is quite limited. More data must be recorded for future research and a more detailed evaluation. Especially in the context of the whole software stack, this will reveal novel points for improvement that can be implemented in future works. Therefore, the performance of the localization module will not be investigated isolated, but the effects of the localization behavior on the rest of the software can be analyzed.

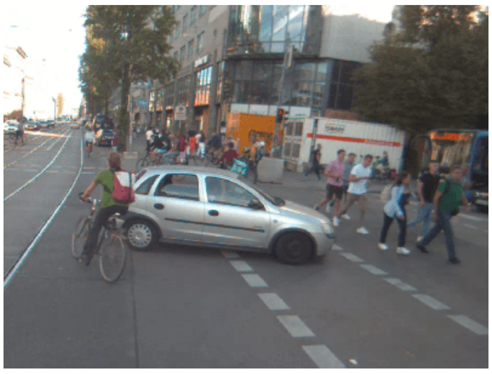
Mapping:

By the improved sensor setup (PTP synchronization, four LiDARs, calibration), significant improvements from the *Garching* dataset to the *Wiesn* dataset could already be observed. However, in the *Wiesn* dataset, the frame rate of the *Ouster* LiDARs dropped, leading to a restricted mapping performance, especially in terms of the density of the map on the sides of the road. The cause of this problem lies within the vehicle platform. Higher framerates of the LiDARs will improve results and denser 3D point cloud maps.

As for the EDGAR research vehicle, existing SLAM approaches failed to create satisfactory maps with all LiDARs, the *KISS-ICP* [82] was adapted to handle all four laser scanners and combine their point clouds. However, it is only a LiDAR odometry, meaning it does not provide any loop closure, global optimization, or GNSS integration capability. To minimize the adverse effects of the accumulated drift, after the *KISS-ICP* mapping, *Interactive* SLAM [317] was applied to manually add loop closure constraints and remove erroneous constraints from the odometry. Therefore, manual steps are required, which also induce additional uncertainty. In the desirable mapping pipeline, a SLAM back-end would be added to the *KISS-ICP* front-end. By this, GNSS and IMU measurements could be directly included, output drift-compensated, and even georeferenced trajectories and maps without manual post-processing.

To obtain georeferenced maps without this back-end, the *FlexMap Fusion* tool was developed. It uses GNSS data to georeference the created map a posteriori. The rubber sheeting algorithm used currently only works two-dimensional, meaning that height drift cannot be compensated. Here, two suggestions for improvement have to be made. First, correspondences between the *KISS-ICP* and the GNSS trajectory need to be found based on their timestamps. By this, no manual control points need to be selected, leading to decreased manual effort and uncertainty. Therefore, the *Interactive* SLAM needs to be adapted to save the timestamps of all point clouds after processing. Second, extending the rubber-sheeting to three dimensions is desirable for future applications. This allows to remove the drift in z-direction and align the point cloud map and the lanelet map in all three dimensions. Therefore, the triangulation needs to be extended to divide the corresponding volume into tetrahedrons and interpolate the control point errors in between.

A general challenge for mapping for autonomous driving is handling dynamic objects. By design, SLAM algorithms work under the Markov assumption, assuming a static environment, and do not remove dynamic objects from this assumption. In a case with many moving objects, e.g., at crowded corners in cities, dynamic objects can even lead to mistracking in the mapping process. This behavior can be seen in Figure 5.19. In this scene, the ego vehicle was waiting to turn right and cross the pedestrian crosswalk. As the ego was not moving, but large parts of the surrounding (vehicles, pedestrians, cyclists) were, LiDAR registration failed, and a vehicle movement was estimated (Figure 5.19 (b)). In the case of a standing ego vehicle, this could be detected, for example, by considering wheel odometry and triggering a *standstill* mode. However, similar cases can also appear on crowded and multi-lane roads when the ego vehicle moves. This makes it hard to accurately map busy roads like the *Mittlerer Ring* in Munich.


(a) Corresponding camera image showing the crowded and busy intersection.

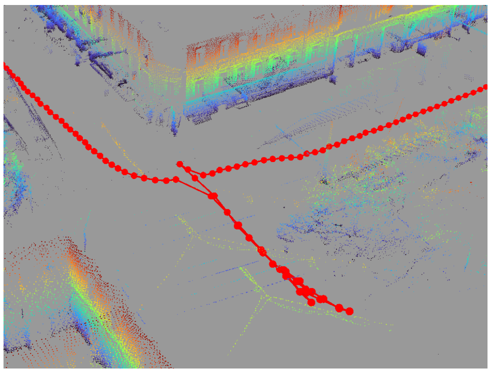
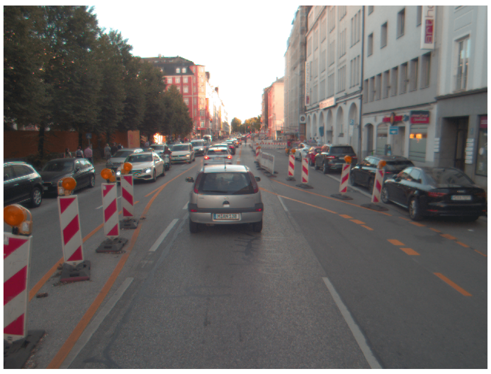

(b) Estimated trajectory from the *KISS-ICP*, loaded into *Interactive* SLAM.

Figure 5.19: Section from the *Wiesn* dataset where *KISS-ICP* struggles to estimate the correct trajectory.

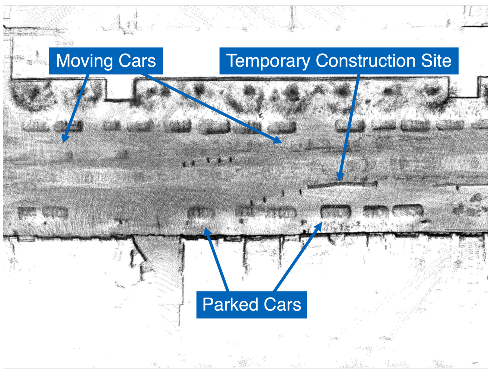
In addition to these dynamic objects, some are static during mapping but can appear or disappear between mapping and driving the mapped area. This could be, for example, parked cars, construction sites, etc. Figure 5.20 shows an excerpt from the *Wiesn* dataset at a temporal construction site. There are temporary yellow-marked lanes that lead through the opposite lanes. These lanes will occur in the point cloud map but not in the OSM based lanelet map. Also, this construction site might be decommissioned at the time of driving through this region again and trying to use the map.

To solve this challenge, there are a few solutions, as already described in Subsection 2.3.2. A greater number of (human-driven) vehicles can be leveraged to detect and even remap such areas. These need to be able to detect such map changes with sensors and collect the information. Another option is to detect and handle these situations online while driving autonomously. However, this requires either a software stack capable of driving without HD maps or a fallback, such as teleoperation.

Regarding semantic map generation, the presented pipeline mainly uses publicly and crowd-sourced data OSM. These data showed to be significantly more detailed and up-to-date than *Google Maps*, for example.



(a) Camera image from the *Wiesn* dataset. There are temporary lanes due to a construction site.



(b) Corresponding scene in the point cloud map.

Figure 5.20: Section from the *Wiesn* dataset showing a construction site, temporary lanes, and parked and driving cars.

However, as the data only comprise the centerline of the road and the number of lanes, heuristic assumptions had to be made to generate a lanelet map including all lanes and their boundaries. This worked sufficiently well for the two presented datasets; however, it relies on the `highway` tag from the data and induces uncertainty. Also, this process can cause problems, as lane merges or intersections are not modeled as detailed as needed. Another limitation of OSM data is that most of it does not contain altitude data for three-dimensional map matching. During the creation of the two presented HD maps, it was observed that the quality of the OSM data in the center of Munich was significantly better than at the Garching Campus. This can lead to restrictions when applying the mapping pipeline to rural regions. Modeling intersections, especially with several lanes and turn-off options, poses another challenge. Here, the heuristics in the OSM to *Lanelet2* module fail. This leads to the need for manual post-processing using the *VectorMapBuilder*[1] tool. This process is not viable for scaling the mapping further and being able to create maps of whole cities. Future work should investigate enhanced algorithms to handle lanelets better.

Conclusion:

In summary, this chapter presented the design of an HD mapping pipeline that allows rapid mapping of new regions with limited manual processing. In contrast to available approaches presented in Subsection 2.3.3, an entire pipeline was developed from raw data to usable HD maps.

The presented approach creates a 3D point cloud map and a semantic *Lanelet2* map, which are aligned and georeferenced. The point cloud maps are generated using the LiDAR sensors. Offline postprocessing globally optimizes the generated point cloud. To generate semantic lanelet maps, OSM was used as the primary data source. A novel tool was developed for georeferencing and fusion of the lanelet and point cloud map. This should be done online in future work by directly integrating the GNSS data during the mapping run. A SLAM framework is desirable, integrating the robust *KISS-ICP* solution with a full back-end.

The accuracy of the maps was found sufficient for autonomous driving with human supervision. However, due to dynamic objects and changes in road layout, etc., it is currently impossible to create reliable HD maps in every situation. Local detection of drivable space is needed to detect map changes and the possibility of driving on roads with map errors. The presented pipeline is to be seen as a baseline for future research and enhancements. It provides the capability to create and use maps of previously unmapped regions with full *Autoware* integration. Future research will extend this approach and add more functionalities and novel algorithms consecutively. The following chapter will discuss the future of HD mapping for autonomous driving in more detail.

[1] https://tools.tier4.jp/feature/vector_map_builder_ll2/

6 Discussion and Outlook

After presenting the approaches developed, their results, and the associated discussions individually in Chapter 4 and Chapter 5, this chapter puts them into context with the research questions defined in Chapter 3. On this basis, a comprehensive discussion of the entire work is provided. Extending these findings, the limitations of the presented approaches are elaborated. These lead to future research topics and a general outlook on perception, localization, and mapping for AVs.

6.1 Review of the Research Questions

Before providing a response to the main research question, the two subquestions will be answered. Therefore, algorithms were developed for two real-world applications: the AV-21 for autonomous racing and the EDGAR research vehicle for public traffic. Autonomous racing represented a significant challenge due to high velocities and feature-poor environments. For the EDGAR research vehicle, the focus was shifted toward HD mapping and how it can be applied to this project, the specific vehicle, and realized in a scalable way.

SQ1: How can sensor fusion be used for localization and mapping algorithms in difficult real-world applications?

In Chapter 4, different methods were presented that make use of sensor fusion for the task of localization. As existing algorithms were unable to solve the localization for the AV-21 on oval race tracks, a specific approach was developed that fuses data from camera and LiDAR. Camera data were taken for the longitudinal and LiDAR data for the lateral localization of the vehicle on the race track. This concept allows to benefit from the individual advantages of different sensor modalities. Sensor fusion allowed the development of a working localization concept that was validated in simulation. However, the concept is specific and thus limited to this application domain. Based on the findings, the *ORB-SLAM3 RGB-L* [14, 15] was implemented that uses a well-known visual SLAM framework and adds LiDAR measurements as upsampled depth maps. This approach was shown to improve the performance of the visual SLAM both in terms of accuracy and runtime, especially in environments with a limited amount of features. However, in some scenarios, the performance suffered. Furthermore, this approach increases the requirements for the sensor setup, calibration, and triggering. By the implementation of *CamRaDepth* [16, 17], a more enhanced method of pixel-wise depth map generation based on deep learning transformer architecture was implemented and validated. It showed that through a purposeful training process, depth measurements do not necessarily have to be available at inference time for accurate depth maps. This can lead to decreased requirements for the sensor setup. However, the approach benefitted significantly from the added RaDAR data. Also, adding semantic segmentation as an additional modality improved the results. Finally, *CaLiMO* was introduced and showed potential for future work. It is a SLAM concept that directly fuses information from LiDAR, camera, and IMU in a graph-based back-end.

These approaches proved that sensor fusion could be used to increase the performance of localization algorithms, especially in challenging environments, e.g., feature-poor environments. In some applications, such as autonomous oval racing, reliable localization is only possible through the use of sensor fusion. However, it showed that these approaches tend to be specific and tailored for an ODD and use case. Future approaches should try to use all available information and let the algorithm decide on what data to use in what situation. Therefore, factor graphs offer a great possibility to optimize the whole trajectory graph at runtime. Another challenge for sensor fusion in localization is the sensor setup. To be able to fuse sensor data meaningfully, internal and external calibration, time synchronization, and, for some approaches, simultaneous triggering are crucial. This leads to higher requirements for the robot platform.

To summarize and briefly answer the research question, sensor fusion can help overcome the limitations of localization and mapping algorithms, especially in challenging applications. However, it leads to higher data requirements and induces novel challenges such as calibration, time synchronization, etc. These can limit the transferability of algorithms as seen in Chapter 4. Multimodal algorithms that generalize better to other domains are a desirable topic of future research.

SQ2: How to design an open HD mapping pipeline for AVs?

In Chapter 5, a localization and mapping pipeline for the research vehicle EDGAR was developed and validated. It is compatible with the *Autoware* framework and is mainly built on LiDARs point clouds, GNSS data, and OSM lane information. In comparably feature-rich environments, e.g., in urban scenarios, LiDARs can offer a robust data source for localization and mapping. The main problem stemmed from the sensor setup with errors in the spatial calibration, no triggered sensors, and partially missing temporal synchronization. Therefore, a novel LiDAR fusion approach based on *KISS-ICP* [18, 82] was introduced. As an additional data source, crowd-sourced data from OSM were used for semantic information extraction, which is hard to get from on-board LiDAR data. These provided advantages over semantic maps from sensor data regarding accuracy and expense. The map was designed in a way that, in the future, additional sensors and map layers, such as cameras, can be integrated. The developed map post-processing and fusion tool can be used to align the maps and georeference them. This allows the comparison and benchmarking of different combinations of maps, used sensors, algorithms, etc. The presented mapping approach allows for quick mapping of previously unmapped areas. However, manual processing steps are still required throughout the process. Additionally, slight drifts could still be observed, even after the georeferencing step. The approach proposed in the previous section can also be helpful to reduce the drift. The globally optimal estimated trajectory can be determined if all information, also GNSS, is directly used at map creation. In the current approach, information is lost as the *FlexMap Fusion* tool does not use the entire graph built during the mapping process. Another challenge will be the detection and handling of map changes, which has to be addressed in future work. Currently, the fallback solution for the EDGAR research vehicle is a remote teleoperator that can take over the vehicle.

Summing up, Chapter 5 developed an applicable mapping pipeline from sensor data to HD maps, which showed efficient for the research vehicle EDGAR. This pipeline allows the mapping of spatially limited regions with reasonable manual effort. However, to allow the mapping of increasingly larger areas, the manual steps need to be removed in the future. Also, automated map validation and quality assessment are topics for future research.

RQ: How to formulate an open generic framework for robust and applicable HD mapping and localization for AVs?

Within the scope of this thesis, different approaches for enhanced localization have been introduced and evaluated. Also, a scalable HD mapping pipeline was presented and validated with the research vehicle EDGAR. This thesis dealt with two different use cases: urban autonomous driving and autonomous racing on high-speed tracks. As both projects will be continued in the future, it is desirable to build the algorithms and the code base so that functionalities can be used with both platforms. Therefore, the *core functionality* was

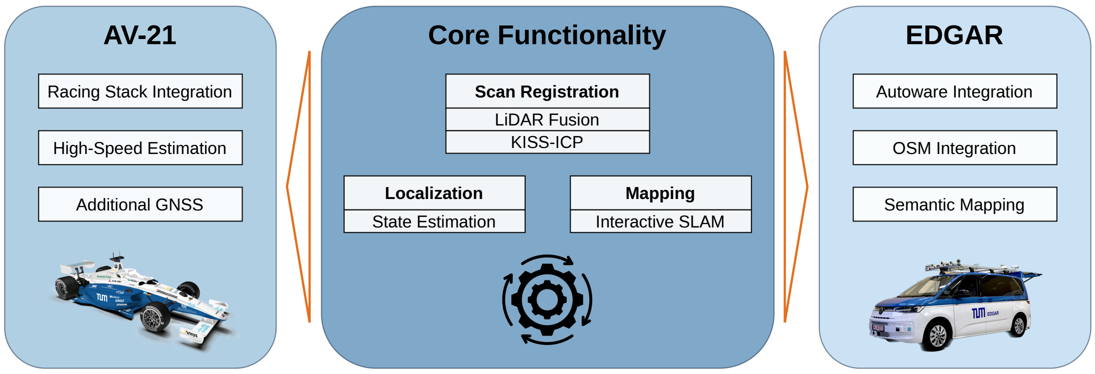


Figure 6.1: Proposed split of the algorithms. The core repository includes the main functionalities. The application is done in the specific repositories.

extracted from the specific application code bases. It includes the multi-LiDAR scan registration, a basic state estimation implementation, and the mapping postprocessing pipeline. These tools are needed for localization and mapping of both race tracks and urban areas.

Figure 6.1 visualizes the project structure. The code is divided into three repositories. One core repository contains and abstracts the described main functionalities for scan registration, localization, and mapping. These are designed in a way to work with different platforms (Section 5.3). The algorithms are used by two application-specific repositories for the AV-21 and EDGAR. This enables at least partial differentiation between the main algorithms and application-specific adjustments. The current, mainly LiDAR-based localization has been proven to work on street-course race tracks and urban areas. For oval race tracks, currently, the state estimation mainly focuses on GNSS localization. Here, future work can focus on including additional localization layers in the framework, such as the one presented in Subsection 4.1.3. However, the limitation of computing resources will be an additional challenge to handle. The provided structure is the base for future research. It allows a quick adaption of algorithms and applications to different use cases. The abstraction of the core functionality allows for quick integration into new vehicle platforms.

In this way, algorithms can be developed and directly evaluated for two different ODDs. The structure can be seen as a framework for future research in localization and mapping. It will support the development of applicable, well-generalizing algorithms with limited overfitting to a specific application.

6.2 Review of the Applications and Future Recommendations

This section reviews the outcomes of this thesis, focusing on the implications for the two key applications: autonomous racing and HD mapping the autonomous research vehicle EDGAR. Based on the findings, recommendations for the future will be given.

Autonomous Racing

For autonomous racing, precise maps provide a huge advantage and are heavily recommended. They reduce online computation as the track boundaries do not have to be extracted from sensor data. Also, map usage allows the software stack to react to track sections out of the perception range. Following, the vehicle can e.g. drive at higher speeds on straights as the next turn is already known hundreds of meters in advance.

Chapter 4 presented approaches to overcome some of the limitations of localization for autonomous racing. However, for the final races at the IMS and LVMS, the main source of localization was the redundant RTK-corrected GNSS system that provides cm-level accuracy under open sky. The GNSS signals were fused in an EKF-based state estimation to provide precise, robust, and high-frequent information on the current vehicle state needed for trajectory planning and control. Besides the limited implementation effort of this approach, another advantage is the small amount of computing power needed. This leaves more computational power for the other modules of the autonomous racing software stack. Also, as a map, only the track boundaries need to be extracted and no map for localization has to be built. This can safe lots of track time which can be used to increase the performance of the remaining software modules. Moreover, GNSS systems have less struggle with high velocities compared to cameras and LiDARs, such as distortion. GNSS is not dependent on environmental features that are difficult to robustly detect and match on oval race tracks. Thus, also for future racing applications under open sky, where good GNSS coverage can be assumed, this is a well-working and proven concept for localization. The state estimation can help the race car to safe-stop in case of GNSS failure.

However, localization solutions also need to be found for race tracks that do not have full GNSS coverage. Also there, it is advised to use GNSS data when available and fuse it into the state estimation. This needs to track the quality of the signal and rely on other localization sources in areas where the GNSS performance decreases. When the GNSS signals are blocked, this usually means that there are obstacles surrounding the track that shield the satellite signals. These obstacles usually provide features that can be used for LiDAR or camera localization, leading to a synergy between GNSS in open, feature-poor regions and LiDAR or camera-based localization in GNSS-denied regions of the track. Here, the LiDAR approach presented in Subsubsection 4.1.2 and Section 5.3, represents a good starting point for future research. It allows to offline build 3D point cloud maps that can be used for online LiDAR localization. The tool presented in Subsection 5.4.4 can be used for georeferencing of the created maps. This allows a fusion of GNSS and LiDAR localization within the state estimation.

Data from different race tracks with different layouts, environments, and characteristics will have to be collected to enhance this approach, make it robust, and validate the results. This opens up great possibilities for future research.

EDGAR Research Vehicle

For the EDGAR research vehicle, it is advised to use and create synergies by building upon the same core concept as autonomous racing. This was already described in the previous section. Therefore, the LiDAR mapping and localization approach was adapted to work with the specific sensor and multi-LiDAR setup of the EDGAR research vehicle. To allow urban autonomous driving, this approach has to be enhanced with several additions, such as mapping of road lanes, traffic lights and signs, crosswalks, etc. Therefore, OSM provides a good database that can be used and enhanced with detected objects. To provide a good research base and allow the evaluation of different algorithms, the mapping pipeline needs to stick to open standards, such as the *Lanelet2* format.

This thesis presented the first implementation of this open mapping research pipeline. However, there are still some drawbacks that need to be solved in future research. First of all, the map accuracy needs to be improved. The HD maps in this thesis showed accuracies in the region of $10\,\mathrm{cm}$ to $40\,\mathrm{cm}$, with maximum errors exceeding $1\,\mathrm{m}$. The desired accuracy is below $20\,\mathrm{cm}$. Therefore, the handling of dynamic objects is a crucial step and needs to be addressed. In contrast to racing, it is not possible to do dedicated mapping runs without other road users.

Another topic that needs to be addressed in future work is the awareness of map accuracy. In areas with lower accuracy, e.g. through construction sites, the driving style needs to be adapted and the vehicle needs to rely more on sensor data to reduce the risk. Currently, during online localization, the map is always used in the same way without monitoring the accuracy and reliability. To realize this, the software stack needs to be aware of the map quality in different regions.

6.3 From Race Track to Public Roads

This thesis has presented different approaches for localization and mapping on race tracks and public roads. As described before, these algorithms had to be specifically tailored to their ODD to obtain convincing results. Thus, the question arises of what a transfer from the race track to public roads and vice versa can look like. The previous section already presented a concept that divides between core functionality and application-specific additions. This allows the use of the algorithms for both use cases and adds platform-specific layers for each vehicle. However, in the current state, not all the localization concepts presented in Chapter 4 are included in this framework. The thesis proved that many of the findings of autonomous racing can be transferred to the localization on public roads. Mainly, these are:

- Handling and fusion approach of multiple LiDAR sensors.
- Dependence on the sensor setup (calibration, time synchronization, triggering).
- The 3D point cloud mapping approach.

Nevertheless, it was not possible to transfer all developed algorithms directly due to integration effort and applicability. Future research will have to be performed to further decouple the hardware and sensor setup of the vehicle from the localization functionality. Algorithms that make maximum use of contained information in sensor data are needed. In Subsection 4.1.3, this was achieved by explicitly splitting the tasks in the localization algorithm according to the sensor data. Future approaches should implicitly manage sensor data more intelligently to allow for comparable results at better generalization. This enables easy adaption to other sensor setups and can also handle sensor failure at runtime. New sensors can then be added to this approach.

Future research will have to focus on generalization from one domain to another. This will not only support the transfer from race tracks to public roads but also expand the applicable ODDs with the final goal of reaching ODD independent level 5 autonomy. A significant challenge towards reaching this goal is the data base for vehicle evaluation. This will be addressed in the next section.

6.4 Data Dependency and Shortcomings of Available Datasets

Even though the algorithms presented in this thesis, and generally SLAM-based approaches, mainly do not rely on data-driven deep learning approaches, it has turned out that they are still heavily dependent on the data used for development. Thus, diverse and comprehensive datasets are necessary to implement algorithms working across different ODDs. Diversity relates to different aspects of the data and needs to be multi-dimensional. Mainly, the following characteristics of the data can be varied:

- Geometric properties of the environment
- Sensor setup of the ego vehicle
- Driven trajectory, loop closure events, distance
- Driving profile: accelerations, velocities, yaw rates, etc.

- Weather and lighting conditions
- Number and density of surrounding traffic

Especially in terms of datasets with a multi-LiDAR setup, a deficiency was determined, leading to a lack of algorithms capable of handling the EDGAR LiDAR setup. Additionally, many algorithms incorporate prior knowledge or assumptions about the sensor positioning or the environment, e.g., a $360°$ LiDAR mounted parallel to the floor is assumed [94]. If the algorithm is tested on multiple datasets and multiple sensor configurations, such assumptions might not be valid, forcing algorithms to show better generalizing behavior. Similar observations can be made for weather conditions, scenery etc. For example, many algorithms struggle with the highway scenarios of the *KITTI* dataset as they were developed for urban areas. In our specific case, the *KISS-ICP* showed the most convincing results not because it was the most enhanced algorithm but because it was developed for different applications and ODDs.

Many existing datasets depict an optimal world with synchronized and triggered sensors, de-skewed point clouds, precise calibration data, etc. In real-world applications, some of these advantages might not be applicable, leading to different behavior of the algorithms and worse pose tracking capabilities. A wide variety of different and diverse datasets is the foundation for level 5 autonomy. This will also be required when it comes to validation, safety, and approval, a crucial step on the way to autonomy on public roads [352]. Diverse datasets can help ensure that SLAM algorithms meet strict safety requirements for deployment on public roads.

Although traditional SLAM algorithms are not inherently data-driven, there is research on integrating machine learning algorithms into conventional approaches. Integration of machine learning for components such as feature extraction and matching or semantic understanding benefits from diverse datasets. The increasing trend towards hybrid systems that integrate traditional SLAM with machine learning methods requires a variety of datasets to train and optimize these models efficiently [353].

In summary, the need for more diverse datasets for autonomous vehicle localization is clear. If one dataset includes diverse data as described above, algorithms can be directly tested without the need to adapt interfaces, etc. Even for algorithms that are not inherently data-driven, the benefits of having diverse and extensive datasets are numerous. They ensure that algorithms are robust, adaptable, and safe for real-world applications, meeting the needs of increasingly complex ODDs for autonomous driving. The recently published *HeLiPR* dataset [354] puts its focus on these topics. It offers four different LiDAR sensors and spatial and temporal variance for different challenges in each sequence. This dataset is an excellent evaluation and testing base for future algorithms.

6.5 Solution to Scalability and the Future of HD-Maps

Chapter 5 presented the development and realization of an HD mapping pipeline. The focus was to develop a scalable approach to quickly add new, previously unmapped areas without the need for dedicated mapping vehicles. Therefore, a point cloud mapping approach using on-board LiDARs was combined with a semantic mapping approach using publicly available data from OSM. Manual refinement steps were necessary for both map creation steps to generate consistent maps suitable for fully autonomous driving. For the point cloud map, loop constraints were added, and faulty parts of the graph were manually removed. For this problem, a solution has already been presented: The current approach needs to be extended with a back-end that combines measurements from all sensors, including GNSS. Thus, the subsequent georeferencing step would also be obsolete. Manual refinement was also necessary to generate lanelet maps to ensure the consistency and correct assignment of all lanes. Complex lane merges and crossings pose problems especially. With the pipeline presented, valid HD maps in the order of the size of those presented in Section 5.5 can be created within a day from data recording. However, the pipeline is not suitable for creating maps of entire cities, such as Munich. Even if all manual steps could be removed, the scalability problem would not finally be solved. One big problem remaining is the map validation. This could be done for the EDGAR project by driving through the mapped areas and validating the map under human supervision. Autonomous vehicles must rely on the map without human validation for level 4 or even level 5 autonomy in large cities. Another problem that was already discussed in Section 5.5 is the change of the area. Especially in big cities, the environment, including the geometric structure and the road layout, can change from day to day. The presented approach cannot detect map changes or handle them at runtime. This is an ongoing research topic. Solution ideas have been presented in Subsection 2.3.2. If the detection of map changes and the correction of the existing map are solved, the handling of the whole map data needs to be addressed. Large cities' maps will reach sizes too large to be entirely stored in each vehicle. Therefore, the maps will have to be distributed to each car via a mobile network. This requires intelligent algorithms for map handling and dynamically uploading and downloading updates.

To reach full level 5 autonomy, autonomous vehicles must work even if HD maps are not available or faulty. This means that even if HD maps can improve perception capabilities, they must not be required for autonomous driving at some point. If the maps are only taken as an additional input to sensor data, their level of detail can be significantly reduced, also reducing all of the above-mentioned problems. The map could be used and handled as sensor data, accounting for included uncertainty. On the other hand, one could scrutinize the usage of HD maps at all if they are not required. There is ongoing research in the field of online mapping (Subsubsection 2.3.3). This approach uses sensor data to map the environment around the vehicle without using offline generated HD map data. Currently, this approach lags significantly behind offline maps in terms of accuracy; however, it promises enormous potential for future research [261]. Section 6.6 will present a concept for integrating online mapping into an end-to-end perception pipeline for future autonomous vehicles.

6.6 Outlook: Towards End-to-End Perception and Local Mapping

In the final section of this thesis, a deep learning-based concept for future perception systems of AVs will be presented. A first draft of the concept called *DeepSTEP* has already been published [20].

This concept works without offline generated data, such as an HD map, and is based solely on online sensor data from different sensor modalities. Figure 6.2 depicts a generalized architecture of the approach. The multimodal sensor data on the left side can be from LiDARs, cameras, RaDARs, etc. Here, map data can also be encoded into the shared feature space for the usage of prebuilt maps. Sensor-individual feature extractors encode these data into a common representation. As the different modalities have different data representations (e.g., 2D vs. 3D), they all have to be transformed into a shared space, which can be, for example, BEV or a 3D voxel representation.

The heart of this approach is the spatio-temporal fusion network. As a lot of information is contained in the temporal context, e.g., cars cannot disappear within parts of a second even if they are occluded, this concept makes use not only of the spatial but also the temporal relations encoded in the sensor data. The network design aims to contain all the environmental information in the latent space. Different task-specific decoders will use this for object detection, local mapping, etc. In addition to the temporal context, this is another benefit of this approach: The different perception tasks are not uncorrelated (e.g., the probability of detecting cars in a particular area and detecting a road in the same location), so they can benefit from one another. Conventional perception approaches run fully detached object detection and local mapping

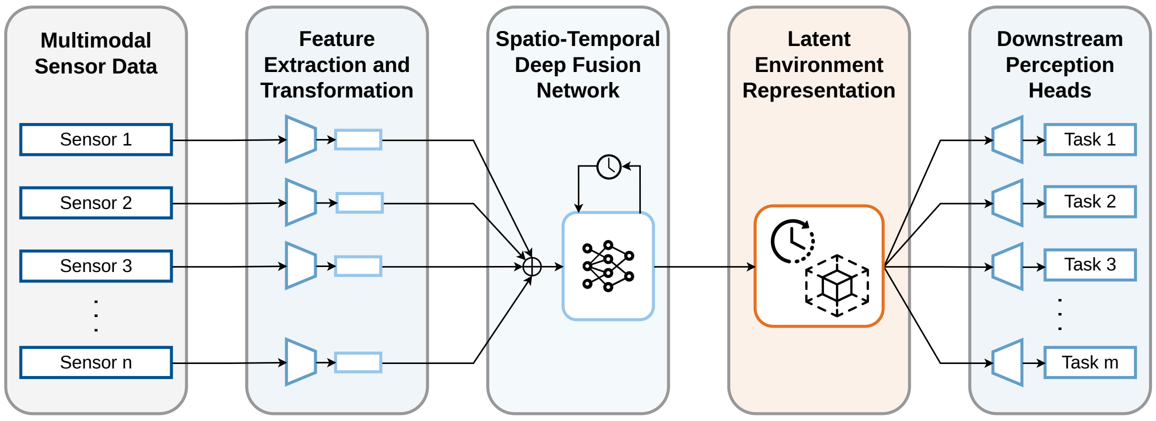

Multimodal Sensor Data
Sensor 1
Sensor 2
Sensor 3
Sensor n
Feature Extraction and Transformation
Spatio-Temporal Deep Fusion Network
Latent Environment Representation
Downstream Perception Heads
Task 1
Task 2
Task 3
Task m

Figure 6.2: Proposed end-to-end perception approach. Adapted from [20].

algorithms. Combining them will make the network more efficient regarding data, runtime, and compute requirements. As only one network is to be deployed, only one network must be trained and executed online while driving.

In summary, the concept of *DeepSTEP* offers excellent possibilities for future research. Mainly, it can improve current perception pipelines through the following three advantages:

- Incorporation of the temporal domain into the latent environment representation.
- The shared feature space for all perception tasks provides benefits for accuracy, runtime, and data efficiency.
- The concept can be arbitrarily expanded on both ends by adding new sensors or additional perception heads.

7 Summary

This thesis dealt with the applied localization and mapping of AVs in different conditions and use cases. Localization and mapping were identified as crucial components of AD software stacks. However, there is still a lack of available algorithms, especially when it comes to various applications. As different sensor modalities and other data sources, such as crowd-sourced maps, have distinct advantages and disadvantages, this thesis investigated the hypothesis that sensor fusion can help overcome some of the limitations. Therefore, algorithms were presented mainly for localization with multimodal data. Based on the findings, a localization and HD mapping pipeline for urban areas was developed and validated.

The introductory chapter motivated the work and stressed its importance. Required general background information on autonomous vehicles, sensors, and software components was provided. Solutions had to be found for the localization of an autonomous race car and the HD map creation of an autonomous shuttle.

The related work (Chapter 2) started with the state of the art of environmental perception. The sensors used were presented, and their models and mathematical formulations were derived. The final section presented an overview of existing datasets that can be used for algorithm development and validation. Using these sensors, various approaches to solving the SLAM problem were presented based on different sensor modalities. These algorithms are used for the generation of the HD maps, which are used in most current approaches. However, many challenges related to HD maps were derived, especially with regard to their scalability. Chapter 3 concluded the related work and derived the research questions that this thesis tried to answer:
How to formulate an open generic framework for robust and applicable HD mapping and localization for AVs?
To answer this question, two subquestions are to be answered:
How can sensor fusion be used for localization and mapping algorithms in difficult real-world applications?
How to design an open HD mapping pipeline for AVs?
The first subquestion was answered by Chapter 4. For the application of multimodal localization algorithms, the vehicle platform AV-21, a fully autonomous race car, was used. As existing localization algorithms failed under challenging conditions, novel approaches that took advantage of sensor fusion were presented. First, LiDAR and camera data were combined in a novel localization algorithm. Therefore, camera images were used to determine the longitudinal and LiDAR point clouds were used to determine the lateral track position [11]. As this approach of combining depth measurements with camera images showed convincing results, it was extended to a more general approach: *ORB-SLAM3 RGB-L*. LiDAR point clouds were projected into the camera image and upsampled to correlate pixels with depth information. This approach was able to outperform the stereo *ORB-SLAM3* [124] in feature-poor environments and in terms of runtime [14]. *CamRaDepth* [16], a deep-learning based approach for pixel-wise depth estimation without LiDAR data was developed that outperformed state-of-the-art algorithms. However, it could not improve the *ORB-SLAM3 RGB-L* over upsampling with conventional CV algorithms. More data are needed to obtain better results. Finally, *CaLiMO* was presented, a SLAM approach using data from camera and LiDAR. First results showed to be promising; however, more work needs to be put into this approach to reach the state of the art.

In Chapter 5, all findings were transferred to the EDGAR research vehicle, to answer the second subquestion. This platform posed challenges through the multi-LiDAR setup with faulty calibrations and partially missing time synchronization. Therefore, a new scan matching approach was developed. This was extended to an offline 3D mapping and an online localization module, both compatible with the *Autoware* base software stack. For semantic lane mapping, OSM data were used, and a novel tool was presented to fuse all map data. The pipeline was validated on the TUM Campus in Garching and a scenario in the city center of Munich. It was possible to build valid and suitable HD maps for autonomous driving. However, few manual post-processing steps were needed to remove map errors. Moreover, dynamic and semi-dynamic objects still pose unsolved challenges for map creation [18]. A quantified validation and assessment of the map quality and accuracy was identified as another important topic that needs to be addressed. The presented open mapping pipeline builds a baseline for future research in the field of HD map generation.

Chapter 6 discussed all of the presented approaches comprehensively. This thesis introduced novel algorithms that can solve some of the existing problems in autonomous vehicle localization and mapping. Nevertheless, there are still many open points, especially the generalization between different vehicle platforms, weather conditions, geometric environments, etc. To address this issue, a framework was presented to abstract core functionality and decouple it from the vehicle-specific integrations. Another major problem is the applicability of HD maps in terms of generation, deployment, and correction. This led to the question of whether HD maps will be the future of autonomous driving. To reach level 5 autonomy, an AV must be able to drive only with the information it receives from the sensors. However, to reach this goal, HD maps are a necessary intermediate step. Finally, an end-to-end perception concept was presented that tries to maximize information usage from sensor data [20].

List of Figures

List of Tables

Bibliography

[1] X. Mosquet, T. Dauner, N. Lang, M. Rüßmann, A. Mei-Pochtler, R. Agrawal and F. Schmieg, „Revolution in the driver's seat: The road to autonomous vehicles,“ *Boston Consulting Group*, vol. 11, 2015. Available: https://www.bcg.com/publications/2015/automotive-consumer-insight-revolution-drivers-seat-road-autonomous-vehicles.

[2] C. F. Forbes. „Cruise Immediately Halts All Robotaxis Nationwide, Seeks To 'Rebuild Trust',“ 2024. [Online]. Available: https://www.forbes.com/sites/cyrusfarivar/2023/10/26/cruise-immediately-halts-all-robotaxis-nationwide-seeks-to-rebuild-trust/ [visited on 02/21/2024].

[3] A. Chougule, V. Chamola, A. Sam, F. R. Yu and B. Sikdar, „A Comprehensive Review on Limitations of Autonomous Driving and Its Impact on Accidents and Collisions,“ *IEEE Open Journal of Vehicular Technology*, vol. 5, pp. 142–161, 2024, DOI: 10.1109/OJVT.2023.3335180.

[4] M. A. Khan, H. E. Sayed, S. Malik, T. Zia, J. Khan, N. Alkaabi and H. Ignatious, „Level-5 autonomous driving—are we there yet? A review of research literature,“ *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–38, 2022, DOI: https://doi.org/10.1145/3485767.

[5] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus and M. H. Ang, „Perception, Planning, Control, and Coordination for Autonomous Vehicles,“ *Machines*, vol. 5, no. 1, 2017, DOI: 10.3390/machines5010006. Available: https://www.mdpi.com/2075-1702/5/1/6.

[6] A. Yuen, J. Murphy, S. Ravipati, D. Goyal, H. Kim, A. Hemakumar, M. Han, A. Ung and C. Corthell. „How Lyft Creates Hyper-Accurate Maps from Open-Source Maps and Real-Time Data,“ 2023. [Online]. Available: https://eng.lyft.com/how-lyft-creates-hyper-accurate-maps-from-open-source-maps-and-real-time-data-8dcf9abdd46a [visited on 11/15/2023].

[7] mobileye. „Road Experience Management,“ 2024. [Online]. Available: https://www.mobileye.com/technology/rem/ [visited on 03/28/2024].

[8] Nvidia. „End-to-end HD Mapping for Self-driving Cars,“ 2024. [Online]. Available: https://www.nvidia.com/en-sg/self-driving-cars/hd-navigation-systems/ [visited on 04/06/2024].

[9] D. Parekh, N. Poddar, A. Rajpurkar, M. Chahal, N. Kumar, G. P. Joshi and W. Cho, „A Review on Autonomous Vehicles: Progress, Methods and Challenges,“ *Electronics*, vol. 11, no. 14, 2022, DOI: 10.3390/electronics11142162. Available: https://www.mdpi.com/2079-9292/11/14/2162.

[10] SAE, „J3016_202104: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,“ 2021.

[11] **F. Sauerbeck**, L. Baierlein, J. Betz and M. Lienkamp, „A Combined LiDAR-Camera Localization for Autonomous Race Cars,“ *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 12-05-01-0006, pp. 61–71, 2022, DOI: https://doi.org/10.4271/12-05-01-0006.

[12] **F. Sauerbeck**, S. Huch, F. Fent, P. Karle, D. Kulmer and J. Betz, „Learn to See Fast: Lessons Learned from Autonomous Racing on How to Develop Perception Systems,“ *IEEE Access*, pp. 1–1, 2023, DOI: https://doi.org/10.1109/ACCESS.2023.3272750.

[13] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp, B. Lohmann, F. Nobis, L. Ögretmen, M. Rowold, **F. Sauerbeck**, T. Stahl, R. Trauth, F. Werner and A. Wischnewski, „TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge,“ *Journal of Field Robotics*, 2022, DOI: https://doi.org/10.1002/rob.22153.

[14] **F. Sauerbeck**, B. Obermeier, M. Rudolph and J. Betz, „RGB-L: Enhancing Indirect Visual SLAM Using LiDAR-Based Dense Depth Maps,“ in *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*, 2023, pp. 95–100, DOI: https://doi.org/10.1109/ICCCR56747.2023.10194045.

[15] **F. Sauerbeck** and M. Rudolph. „GitHub | ORB_SLAM3_RGBL,“ 2023. [Online]. Available: https://github.com/TUMFTM/ORB_SLAM3_RGBL [visited on 07/08/2023].

[16] **F. Sauerbeck**, D. Halperin, L. Connert and J. Betz, „CamRaDepth: Semantic Guided Depth Estimation Using Monocular Camera and Sparse Radar for Automotive Perception,“ *IEEE Sensors Journal*, 2023, DOI: 10.1109/JSEN.2023.3321886.

[17] **F. Sauerbeck** and D. Halperin. „GitHub | CamRaDepth,“ 2023. [Online]. Available: https://github.com/TUMFTM/CamRaDepth [visited on 07/08/2023].

[18] **F. Sauerbeck**, D. Kulmer, M. Leitenstern, C. Weiss and J. Betz, „Multi-LiDAR Localization and Mapping Pipeline for Urban Autonomous Driving,“ (in press), in *2023 IEEE Sensors*, 2023.

[19] M. Leitenstern and **F. Sauerbeck**. „GitHub | Lanelet2_OSM_Fusion,“ 2023. [Online]. Available: https://github.com/TUMFTM/Lanelet2_OSM_Fusion [visited on 10/24/2023].

[20] S. Huch, **F. Sauerbeck** and J. Betz, „DeepSTEP - Deep Learning-Based Spatio-Temporal End-To-End Perception for Autonomous Vehicles,“ in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–8, DOI: https://doi.org/10.1109/IV55152.2023.10186768.

[21] H. Zhu, K.-V. Yuen, L. Mihaylova and H. Leung, „Overview of Environment Perception for Intelligent Vehicles,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2584–2601, 2017, DOI: 10.1109/TITS.2017.2658662.

[22] A. Pandharipande, C.-H. Cheng, J. Dauwels, S. Z. Gurbuz, J. Ibanez-Guzman, G. Li, A. Piazzoni, P. Wang and A. Santra, „Sensing and Machine Learning for Automotive Perception: A Review,“ *IEEE Sensors Journal*, vol. 23, no. 11, pp. 11097–11115, 2023, DOI: 10.1109/JSEN.2023.3262134.

[23] Q. Chen, Y. Xie, S. Guo, J. Bai and Q. Shu, „Sensing system of environmental perception technologies for driverless vehicle: A review of state of the art and challenges,“ *Sensors and Actuators A: Physical*, vol. 319, p. 112566, 2021, DOI: https://doi.org/10.1016/j.sna.2021.112566.

[24] F. Rosique, P. J. Navarro, C. Fernández and A. Padilla, „A systematic review of perception system and simulators for autonomous vehicles research,“ *Sensors*, vol. 19, no. 3, p. 648, 2019, DOI: https://doi.org/10.3390/s19030648.

[25] E. Marti, M. A. De Miguel, F. Garcia and J. Perez, „A review of sensor technologies for perception in automated driving,“ *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 94–108, 2019, DOI: https://10.1109/MITS.2019.2907630.

[26] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy and C. Ryan, „Sensor Technology in Autonomous Vehicles : A review,“ in *2018 29th Irish Signals and Systems Conference (ISSC)*, 2018, pp. 1–4, DOI: 10.1109/ISSC.2018.8585340.

[27] D. J. Yeong, G. Velasco-Hernandez, J. Barry and J. Walsh, „Sensor and sensor fusion technology in autonomous vehicles: A review,“ *Sensors*, vol. 21, no. 6, p. 2140, 2021, DOI: https://doi.org/10.3390/s21062140.

[28] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*, prentice hall professional technical reference, 2002.

[29] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003, DOI: https://doi.org/10.1017/CBO9780511811685.

[30] Z. Zhang, „A flexible new technique for camera calibration,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000, DOI: 10.1109/34.888718.

[31] C. Zhao, Q. Sun, C. Zhang, Y. Tang and F. Qian, „Monocular depth estimation based on deep learning: An overview,“ *Science China Technological Sciences*, vol. 63, no. 9, pp. 1612–1627, 2020.

[32] C. Liu, S. Kumar, S. Gu, R. Timofte and L. Van Gool, „VA-DepthNet: A Variational Approach to Single Image Depth Prediction,“ *arXiv preprint arXiv:2302.06556*, 2023.

[33] C. Liu, S. Kumar, S. Gu, R. Timofte and L. Van Gool, „Single Image Depth Prediction Made Better: A Multivariate Gaussian Take,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17346–17356.

[34] L. Piccinelli, C. Sakaridis and F. Yu, „iDisc: Internal Discretization for Monocular Depth Estimation,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21477–21487.

[35] J. Yan, H. Zhao, P. Bu and Y. Jin, „Channel-wise attention-based network for self-supervised monocular depth estimation,“ in *2021 International Conference on 3D vision (3DV)*, 2021, pp. 464–473.

[36] J. Hu, C. Bao, M. Ozay, C. Fan, Q. Gao, H. Liu and T. L. Lam, „Deep Depth Completion from Extremely Sparse Data: A Survey,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[37] Z. Xie, X. Yu, X. Gao, K. Li and S. Shen, „Recent Advances in Conventional and Deep Learning-Based Depth Completion: A Survey,“ *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[38] J. Ku, A. Harakeh and S. L. Waslander, „In Defense of Classical Image Processing: Fast Depth Completion on the CPU,“ in *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 16–22, DOI: 10.1109/CRV.2018.00013. Available: https://doi.ieeecomputersociety.org/10.1109/CRV.2018.00013.

[39] Y. Zhang, X. Guo, M. Poggi, Z. Zhu, G. Huang and S. Mattoccia, „Completionformer: Depth completion with convolutions and vision transformers,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 18527–18536.

[40] Y. Lin, T. Cheng, Q. Zhong, W. Zhou and H. Yang, „Dynamic spatial propagation network for depth completion,“ in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, pp. 1638–1646.

[41] D. Nazir, A. Pagani, M. Liwicki, D. Stricker and M. Z. Afzal, „SemAttNet: Towards Attention-based Semantic Aware Guided Depth Completion,“ *IEEE Access*, pp. 1–1, 2022, DOI: 10.1109/ACCESS.2022.3214316.

[42] A. Geiger, P. Lenz and R. Urtasun, „Are we ready for autonomous driving? The KITTI vision benchmark suite,“ in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361, DOI: https://10.1109/CVPR.2012.6248074.

[43] D. Soydaner, „Attention mechanism in neural networks: where it comes and where it goes,“ *Neural Computing and Applications*, pp. 1–15, 2022.

[44] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan and J. Hays, „Argoverse: 3d tracking and forecasting with rich maps,“ in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8748–8757, DOI: 10.1109/CVPR.2019.00895.

[45] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan and O. Beijbom, „nuScenes: A Multimodal Dataset for Autonomous Driving,“ in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[46] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen and D. Anguelov, „Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[47] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, T. Fernandez, M. Jänicke, S. Mirashi, C. Savani, M. Sturm, O. Vorobiov, M. Oelker, S. Garreis and P. Schuberth, „A2D2: Audi Autonomous Driving Dataset," 2020, DOI: https://doi.org/10.48550/arXiv.2004.06320. arXiv: 2004.06320 [cs.CV]. Available: https://www.a2d2.audi.

[48] G. Kim, Y. S. Park, Y. Cho, J. Jeong and A. Kim, „MulRan: Multimodal Range Dataset for Urban Place Recognition," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6246–6253, DOI: 10.1109/ICRA40945.2020.9197298.

[49] D. Barnes, M. Gadd, P. Murcutt, P. Newman and I. Posner, „The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6433–6438, DOI: 10.1109/ICRA40945.2020.9196884.

[50] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov and P. Ondruska, „One Thousand and One Hours: Self-driving Motion Prediction Dataset," in *Proceedings of the 2020 Conference on Robot Learning*, 2021, pp. 409–418. Available: https://proceedings.mlr.press/v155/houston21a.html.

[51] P. Xiao, Z. Shao, S. Hao, Z. Zhang, X. Chai, J. Jiao, Z. Li, J. Wu, K. Sun, K. Jiang, Y. Wang and D. Yang, „PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3095–3101, DOI: 10.1109/ITSC48978.2021.9565009.

[52] C. Plachetka, B. Sertolli, J. Fricke, M. Klingner and T. Fingscheidt, „3DHD CityScenes: High-Definition Maps in High-Density Point Clouds," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 627–634, DOI: 10.1109/ITSC55140.2022.9921866.

[53] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. Kaesemodel Pontes, D. Ramanan, P. Carr and J. Hays, „Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/4734ba6f3de83d861c3176a6273cac6d-Paper-round2.pdf.

[54] J. Zhang, H. Zhuge, Y. Liu, G. Peng, Z. Wu, H. Zhang, Q. Lyu, H. Li, C. Zhao, D. Kircali, S. Mharolkar, X. Yang, S. Yi, Y. Wang and D. Wang. *„NTU4DRadLM: 4D Radar-centric Multi-Modal Dataset for Localization and Mapping,"* 2023. arXiv: 2309.00962 [cs.RO].

[55] A. Kulkarni, J. Chrosniak, E. Ducote, **F. Sauerbeck**, A. Saba, U. Chirimar, J. Link, M. Cellina and M. Behl, „RACECAR–The Dataset for High-Speed Autonomous Racing," (in press), in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[56] R. C. Smith and P. Cheeseman, „On the representation and estimation of spatial uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986, DOI: https://doi.org/10.1177/027836498600500404.

[57] H. Durrant-Whyte and T. Bailey, „Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006, DOI: https://10.1109/MRA.2006.1638022.

[58] T. Bailey and H. Durrant-Whyte, „Simultaneous localization and mapping (SLAM): part II,“ *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006, DOI: 10.1109/MRA.2006.1678144.

[59] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani and H. Durrant-Whyte, „Simultaneous localization and mapping with sparse extended information filters,“ *The international journal of robotics research*, vol. 23, no. 7-8, pp. 693–716, 2004, DOI: https://10.1177/0278364904045479.

[60] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert and H. Durrant-Whyte, „Simultaneous localization, mapping and moving object tracking,“ *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007, DOI: https://doi.org/10.1177/0278364907081229.

[61] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, „A solution to the simultaneous localization and map building (SLAM) problem,“ *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 229–241, 2001, DOI: https://10.1109/70.938381.

[62] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J. J. Leonard, „Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age,“ *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016, DOI: 10.1109/TRO.2016.2624754.

[63] F. Dellaert, „Factor graphs and GTSAM: A hands-on introduction,“ *Georgia Institute of Technology, Tech. Rep*, vol. 2, p. 4, 2012.

[64] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, „G2o: A general framework for graph optimization,“ in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613, DOI: 10.1109/ICRA.2011.5979949.

[65] S. Agarwal, K. Mierle and T. C. S. Team. „*Ceres Solver*,“ version 2.2. Oct. 2023. Available: https://github.com/ceres-solver/ceres-solver.

[66] M. Kaess, A. Ranganathan and F. Dellaert, „iSAM: Incremental Smoothing and Mapping,“ *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008, DOI: 10.1109/TRO.2008.2006706.

[67] V. Ila, L. Polok, M. Solony and P. Svoboda, „SLAM++-A highly efficient and temporally scalable incremental SLAM framework,“ *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 210–230, 2017, DOI: https://doi.org/10.1177/0278364917691110.

[68] B. Alsadik and S. Karam, „The simultaneous localization and mapping (SLAM)-An overview,“ *Surv. Geospat. Eng. J*, vol. 2, pp. 34–45, 2021, DOI: https://10.38094/jastt204117.

[69] S. Zheng, J. Wang, C. Rizos, W. Ding and A. El-Mowafy, „Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis,“ *Remote Sensing*, vol. 15, no. 4, p. 1156, 2023, DOI: https://doi.org/10.3390/rs15041156.

[70] T. T. O. Takleh, N. A. Bakar, S. A. Rahman, R. Hamzah and Z. Aziz, „A brief survey on SLAM methods in autonomous vehicle,“ *International Journal of Engineering & Technology*, vol. 7, no. 4, pp. 38–43, 2018, DOI: https://doi.org/10.14419/ijet.v7i4.27.22477.

[71] M. Chghaf, S. Rodriguez and A. E. Ouardi, „Camera, LiDAR and multi-modal SLAM systems for autonomous ground vehicles: a survey,“ *Journal of Intelligent & Robotic Systems*, vol. 105, no. 1, p. 2, 2022, DOI: https://doi.org/10.1007/s10846-022-01582-8.

[72] S. Arshad and G.-W. Kim, „Role of Deep Learning in Loop Closure Detection for Visual and Lidar SLAM: A Survey,“ *Sensors*, vol. 21, no. 4, 2021, DOI: 10.3390/s21041243. Available: https://www.mdpi.com/1424-8220/21/4/1243.

[73] P.-E. Sarlin, D. DeTone, T. Malisiewicz and A. Rabinovich, „SuperGlue: Learning Feature Matching With Graph Neural Networks,“ in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4937–4946, DOI: 10.1109/CVPR42600.2020.00499.

[74] S. Mokssit, D. B. Licea, B. Guermah and M. Ghogho, „Deep Learning Techniques for Visual SLAM: A Survey,“ *IEEE Access*, vol. 11, pp. 20026–20050, 2023, DOI: 10.1109/ACCESS.2023.3249661.

[75] P. J. Besl and N. D. McKay, „Method for registration of 3-D shapes,“ in *Sensor fusion IV: control paradigms and data structures*, 1992, pp. 586–606.

[76] A. Segal, D. Haehnel and S. Thrun, „Generalized-icp.“ in *Robotics: science and systems*, 2009, p. 435.

[77] P. Biber and W. Strasser, „The normal distributions transform: a new approach to laser scan matching,“ in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, 2003, 2743–2748 vol.3, DOI: 10.1109/IROS.2003.1249285.

[78] M. Magnusson, A. Lilienthal and T. Duckett, „Scan registration for autonomous mining vehicles using 3D-NDT,“ *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007, DOI: https://doi.org/10.1002/rob.20204.

[79] J. Behley and C. Stachniss, „Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments.“ in *Robotics: Science and Systems*, 2018, p. 59, DOI: 10.15607/RSS.2018.XIV.016.

[80] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley and C. Stachniss, „SuMa++: Efficient LiDAR-based Semantic SLAM,“ in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537, DOI: 10.1109/IROS40897.2019.8967704.

[81] J.-E. Deschaud, „IMLS-SLAM: Scan-to-Model Matching Based on 3D Data,“ in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2480–2485, DOI: 10.1109/ICRA.2018.8460653.

[82] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley and C. Stachniss, „Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way,“ *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 1029–1036, 2023, DOI: 10.1109/LRA.2023.3236571.

[83] P. Dellenbach, J.-E. Deschaud, B. Jacquet and F. Goulette, „CT-ICP: Real-time elastic LiDAR odometry with loop closure,“ in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5580–5586, DOI: https://10.1109/icra46639.2022.9811849.

[84] X. Zheng and J. Zhu, „Traj-LO: In Defense of LiDAR-Only Odometry Using an Effective Continuous-Time Trajectory,“ *arXiv preprint arXiv:2309.13842*, 2023.

[85] W. Xu, Y. Cai, D. He, J. Lin and F. Zhang, „FAST-LIO2: Fast Direct LiDAR-Inertial Odometry,“ *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022, DOI: 10.1109/TRO.2022.3141876.

[86] W. Xu and F. Zhang, „FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter,“ *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021, DOI: 10.1109/LRA.2021.3064227.

[87] X. Zheng and J. Zhu, „Efficient LiDAR Odometry for Autonomous Driving,“ *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8458–8465, 2021, DOI: 10.1109/LRA.2021.3110372.

[88] Y. Guo, F. Sohel, M. Bennamoun, M. Lu and J. Wan, „Rotational projection statistics for 3D local surface description and object recognition,“ *International journal of computer vision*, vol. 105, pp. 63–86, 2013, DOI: https://doi.org/10.1007/s11263-013-0627-y.

[89] R. B. Rusu, N. Blodow and M. Beetz, „Fast Point Feature Histograms (FPFH) for 3D registration,“ in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217, DOI: 10.1109/ROBOT.2009.5152473.

[90] B. Steder, R. B. Rusu, K. Konolige and W. Burgard, „NARF: 3D range image features for object recognition,“ in *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010, p. 2.

[91] Y. Pan, P. Xiao, Y. He, Z. Shao and Z. Li, „MULLS: Versatile LiDAR SLAM via multi-metric linear least square," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11633–11640, DOI: https://10.1109/ICRA48506.2021.9561364.

[92] J. Zhang and S. Singh, „LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and systems*, 2014, pp. 1–9, DOI: https://doi.org/10.15607/rss.2014.x.007.

[93] H. Wang, C. Wang, C.-L. Chen and L. Xie, „F-LOAM : Fast LiDAR Odometry and Mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4390–4396, DOI: 10.1109/IROS51168.2021.9636655.

[94] T. Shan and B. Englot, „Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4758–4765, DOI: https://10.1109/IROS.2018.8594299.

[95] J. Lin and F. Zhang, „Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3126–3131, DOI: 10.1109/ICRA40945.2020.9197440.

[96] J. Jiao, H. Ye, Y. Zhu and M. Liu, „Robust odometry and mapping for multi-lidar systems with online extrinsic calibration," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 351–371, 2021, DOI: 10.1109/TRO.2021.3078287.

[97] W. Hess, D. Kohler, H. Rapp and D. Andor, „Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278, DOI: 10.1109/ICRA.2016.7487258.

[98] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti and D. Rus, „LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5135–5142, DOI: 10.1109/IROS45743.2020.9341176.

[99] D. Chang, R. Zhang, S. Huang, M. Hu, R. Ding and X. Qin, „WiCRF: Weighted Bimodal Constrained LiDAR Odometry and Mapping With Robust Features," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1423–1430, 2023, DOI: 10.1109/LRA.2022.3233229.

[100] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan and S. Song, „DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 12–21, DOI: 10.1109/ICCV.2019.00010.

[101] M. A. Uy and G. H. Lee, „PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4470–4479, DOI: 10.1109/CVPR.2018.00470.

[102] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart and C. Cadena, „SegMap: 3d segment mapping using data-driven descriptors," *arXiv preprint arXiv:1804.09557*, 2018, DOI: https://doi.org/10.15607/RSS.2018.XIV.003.

[103] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng and J. Li, „LO-Net: Deep Real-Time Lidar Odometry," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8465–8474, DOI: 10.1109/CVPR.2019.00867.

[104] W. Lu, Y. Zhou, G. Wan, S. Hou and S. Song, „L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6382–6391, DOI: 10.1109/CVPR.2019.00655.

[105] M. U. Khan, S. A. A. Zaidi, A. Ishtiaq, S. U. R. Bukhari, S. Samer and A. Farman, „A comparative survey of lidar-slam and lidar based sensor technologies," in *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*, 2021, pp. 1–8, DOI: 10.1109/MAJICC53071.2021.9526266.

[106] M. Kim, M. Zhou, S. Lee and H. Lee, „Development of an Autonomous Mobile Robot in the Outdoor Environments with a Comparative Survey of LiDAR SLAM,“ in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, 2022, pp. 1990–1995, DOI: 10.23919/ICCAS55662.2022.10003762.

[107] D. Tiozzo Fasiolo, L. Scalera, E. Maset and A. Gasparetto, „Experimental evaluation and comparison of LiDAR SLAM algorithms for mobile robotics,“ in *The International Conference of IFToMM ITALY*, 2022, pp. 795–803, DOI: https://doi.org/10.1007/978-3-031-10776-4_91.

[108] D. Van Nam and K. Gon-Woo, „Solid-state LiDAR based-SLAM: A concise review and application,“ in *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2021, pp. 302–305, DOI: 10.1109/BigComp51126.2021.00064.

[109] D. G. Lowe, „Object recognition from local scale-invariant features,“ in *Proceedings of the seventh IEEE international conference on computer vision*, 1999, pp. 1150–1157.

[110] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, „Speeded-Up Robust Features (SURF),“ *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008, DOI: https://doi.org/10.1016/j.cviu.2007.09.014. Available: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

[111] D. G. Viswanathan, „Features from accelerated segment test (fast),“ in *Proceedings of the 10th workshop on image analysis for multimedia interactive services, London, UK*, 2009, pp. 6–8.

[112] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha and P. Fua, „BRIEF: Computing a local binary descriptor very fast,“ *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1281–1298, 2011, DOI: 10.1109/TPAMI.2011.222.

[113] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, „ORB: An efficient alternative to SIFT or SURF,“ in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571, DOI: 10.1109/ICCV.2011.6126544.

[114] S. Leutenegger, M. Chli and R. Y. Siegwart, „BRISK: Binary Robust invariant scalable keypoints,“ in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555, DOI: 10.1109/ICCV.2011.6126542.

[115] J. Shi and Tomasi, „Good features to track,“ in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600, DOI: 10.1109/CVPR.1994.323794.

[116] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel and G. Randall, „LSD: A Fast Line Segment Detector with a False Detection Control,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010, DOI: 10.1109/TPAMI.2008.300.

[117] A. J. Davison, „Real-time simultaneous localisation and mapping with a single camera,“ in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, 1403–1410 vol.2, DOI: 10.1109/ICCV.2003.1238654.

[118] A. J. Davison, I. D. Reid, N. D. Molton and O. Stasse, „MonoSLAM: Real-Time Single Camera SLAM,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007, DOI: 10.1109/TPAMI.2007.1049.

[119] G. Klein and D. Murray, „Parallel Tracking and Mapping for Small AR Workspaces,“ in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234, DOI: 10.1109/ISMAR.2007.4538852.

[120] G. Klein and D. Murray, „Parallel Tracking and Mapping on a camera phone,“ in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 83–86, DOI: 10.1109/ISMAR.2009.5336495.

[121] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige and R. Siegwart, „Keyframe-based visual-inertial slam using nonlinear optimization,“ *Proceedings of Robotis Science and Systems (RSS) 2013*, 2013.

[122] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos, „ORB-SLAM: a versatile and accurate monocular SLAM system,“ *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[123] R. Mur-Artal and J. D. Tardós, „Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,“ *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017, DOI: https://10.1109/TRO.2017.2705103.

[124] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel and J. D. Tardós, „Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,“ *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021, DOI: https://10.1109/TRO.2021.3075644.

[125] S. Sumikura, M. Shibuya and K. Sakurada, „OpenVSLAM: A versatile visual SLAM framework,“ in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2292–2295, DOI: https://doi.org/10.1145/3343031.3350539.

[126] I. Cvišić, J. Ćesić, I. Marković and I. Petrović, „SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles,“ *Journal of field robotics*, vol. 35, no. 4, pp. 578–595, 2018, DOI: https://doi.org/10.1002/rob.21762.

[127] I. Cvišić, I. Marković and I. Petrović, „SOFT2: stereo visual odometry for road vehicles based on a point-to-epipolar-line metric,“ *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 273–288, 2022, DOI: 10.1109/TRO.2022.3188121.

[128] I. Cvišić, I. Marković and I. Petrović, „Recalibrating the KITTI Dataset Camera Setup for Improved Odometry Accuracy,“ in *2021 European Conference on Mobile Robots (ECMR)*, 2021, pp. 1–6, DOI: 10.1109/ECMR50962.2021.9568821.

[129] I. Cvišić, I. Marković and I. Petrović, „Enhanced calibration of camera setups for high-performance visual odometry,“ *Robotics and Autonomous Systems*, vol. 155, p. 104189, 2022, DOI: https://doi.org/10.1016/j.robot.2022.104189.

[130] C. Forster, Z. Zhang, M. Gassner, M. Werlberger and D. Scaramuzza, „SVO: Semidirect visual odometry for monocular and multicamera systems,“ *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016, DOI: https://10.1109/TRO.2016.2623335.

[131] R. A. Newcombe, S. J. Lovegrove and A. J. Davison, „DTAM: Dense tracking and mapping in real-time,“ in *2011 International Conference on Computer Vision*, 2011, pp. 2320–2327, DOI: 10.1109/ICCV.2011.6126513.

[132] T. Pire, T. Fischer, J. Civera, P. De Cristóforis and J. J. Berlles, „Stereo parallel tracking and mapping for robot localization,“ in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1373–1378, DOI: 10.1109/IROS.2015.7353546.

[133] J. Engel, T. Schöps and D. Cremers, „LSD-SLAM: Large-scale direct monocular SLAM,“ in *European conference on computer vision*, 2014, pp. 834–849, DOI: https://doi.org/10.1007/978-3-319-10605-2_54.

[134] A. Concha and J. Civera, „DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence,“ in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5686–5693, DOI: 10.1109/IROS.2015.7354184.

[135] J. Engel, V. Koltun and D. Cremers, „Direct sparse odometry,“ *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017, DOI: https://10.1109/TPAMI.2017.2658577.

[136] R. Wang, M. Schworer and D. Cremers, „Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras,“ in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3903–3911, DOI: https://10.1109/ICCV.2017.421.

[137] G. Pascoe, W. Maddern, M. Tanner, P. Piniés and P. Newman, „NID-SLAM: Robust Monocular SLAM Using Normalised Information Distance,“ in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1446–1455, DOI: 10.1109/CVPR.2017.158.

[138] C. Masone and B. Caputo, „A Survey on Deep Visual Place Recognition,“ *IEEE Access*, vol. 9, pp. 19516–19547, 2021, DOI: 10.1109/ACCESS.2021.3054937.

[139] X. Zhang, L. Wang and Y. Su, „Visual place recognition: A survey from deep learning perspective,“ *Pattern Recognition*, vol. 113, p. 107760, 2021, DOI: https://doi.org/10.1016/j.patcog.2020.107760.

[140] R. Wang, Y. Shen, W. Zuo, S. Zhou and N. Zheng, „TransVPR: Transformer-Based Place Recognition with Multi-Level Attention Aggregation,“ in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 13638–13647, DOI: 10.1109/CVPR52688.2022.01328.

[141] Z. Chen, A. Jacobson, N. Sünderhauf, B. Upcroft, L. Liu, C. Shen, I. Reid and M. Milford, „Deep learning features at scale for visual place recognition,“ in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3223–3230, DOI: 10.1109/ICRA.2017.7989366.

[142] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla and J. Sivic, „NetVLAD: CNN Architecture for Weakly Supervised Place Recognition,“ in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5297–5307, DOI: 10.1109/CVPR.2016.572.

[143] J. McCormac, R. Clark, M. Bloesch, A. Davison and S. Leutenegger, „Fusion++: Volumetric Object-Level SLAM,“ in *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 32–41, DOI: 10.1109/3DV.2018.00015.

[144] N. Yang, L. von Stumberg, R. Wang and D. Cremers, „D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry,“ in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1278–1289, DOI: 10.1109/CVPR42600.2020.00136.

[145] R. Li, S. Wang and D. Gu, „DeepSLAM: A Robust Monocular SLAM System With Unsupervised Deep Learning,“ *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2021, DOI: 10.1109/TIE.2020.2982096.

[146] U.-H. Kim, S.-H. Kim and J.-H. Kim, „SimVODIS: Simultaneous Visual Odometry, Object Detection, and Instance Segmentation,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 428–441, 2022, DOI: 10.1109/TPAMI.2020.3007546.

[147] U.-H. Kim, S.-H. Kim and J.-H. Kim, „SimVODIS++: Neural Semantic Visual Odometry in Dynamic Environments,“ *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4244–4251, 2022, DOI: 10.1109/LRA.2022.3150854.

[148] K. He, G. Gkioxari, P. Dollár and R. Girshick, „Mask R-CNN,“ in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988, DOI: 10.1109/ICCV.2017.322.

[149] K. He, G. Gkioxari, P. Dollár and R. Girshick, „Mask R-CNN,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020, DOI: 10.1109/TPAMI.2018.2844175.

[150] A. Macario Barros, M. Michel, Y. Moline, G. Corre and F. Carrel, „A comprehensive survey of visual slam algorithms,“ *Robotics*, vol. 11, no. 1, p. 24, 2022, DOI: https://doi.org/10.3390/robotics11010024.

[151] I. A. Kazerouni, L. Fitzgerald, G. Dooly and D. Toal, „A survey of state-of-the-art on visual SLAM,“ *Expert Systems with Applications*, vol. 205, p. 117734, 2022, DOI: https://doi.org/10.1016/j.eswa.2022.117734.

[152] W. Chen, G. Shang, A. Ji, C. Zhou, X. Wang, C. Xu, Z. Li and K. Hu, „An overview on visual slam: From tradition to semantic,“ *Remote Sensing*, vol. 14, no. 13, p. 3010, 2022, DOI: https://doi.org/10.3390/rs14133010.

[153] A. Tourani, H. Bavle, J. L. Sanchez-Lopez and H. Voos, „Visual SLAM: what are the current trends and what to expect?,“ *Sensors*, vol. 22, no. 23, p. 9297, 2022, DOI: https://doi.org/10.3390/s22239297.

[154] B. Gao, H. Lang and J. Ren, „Stereo visual SLAM for autonomous vehicles: A review,“ in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 1316–1322, DOI: https://10.1109/SMC42975.2020.9283161.

[155] K. J. de Jesus, H. J. Kobs, A. R. Cukla, M. A. de Souza Leite Cuadros and D. F. T. Gamarra, „Comparison of Visual SLAM Algorithms ORB-SLAM2, RTAB-Map and SPTAM in Internal and External Environments with ROS,“ in *2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE)*, 2021, pp. 216–221, DOI: 10.1109/LARS/SBR/WRE54079.2021.9605432.

[156] J. Cheng, L. Zhang, Q. Chen, X. Hu and J. Cai, „A review of visual SLAM methods for autonomous driving vehicles,“ *Engineering Applications of Artificial Intelligence*, vol. 114, p. 104992, 2022, DOI: https://doi.org/10.1016/j.engappai.2022.104992.

[157] T. Lai, „A Review on Visual-SLAM: Advancements from Geometric Modelling to Learning-Based Semantic Scene Understanding Using Multi-Modal Sensor Fusion,“ *Sensors*, vol. 22, no. 19, p. 7265, 2022, DOI: https://doi.org/10.3390/s22197265.

[158] C. Debeunne and D. Vivet, „A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping,“ *Sensors*, vol. 20, no. 7, 2020, DOI: 10.3390/s20072068. Available: https://www.mdpi.com/1424-8220/20/7/2068.

[159] J. Zhang, M. Kaess and S. Singh, „Real-time depth enhanced monocular odometry,“ in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4973–4980, DOI: 10.1109/IROS.2014.6943269.

[160] J. Zhang, M. Kaess and S. Singh, „A real-time method for depth enhanced visual odometry,“ *Autonomous Robots*, vol. 41, pp. 31–43, 2017, DOI: https://doi.org/10.1007/s10514-015-9525-1.

[161] J. Graeter, A. Wilczynski and M. Lauer, „LIMO: Lidar-Monocular Visual Odometry,“ in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 7872–7879, DOI: 10.1109/IROS.2018.8594394.

[162] Y.-S. Shin, Y. S. Park and A. Kim, „Direct Visual SLAM Using Sparse Depth for Camera-LiDAR System,“ in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5144–5151, DOI: 10.1109/ICRA.2018.8461102.

[163] J. Zhang and S. Singh, „Visual-lidar odometry and mapping: Low-drift, robust, and fast,“ in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2174–2181, DOI: 10.1109/ICRA.2015.7139486.

[164] J. Zhang and S. Singh, „Laser–visual–inertial odometry and mapping with high robustness and low drift,“ *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, 2018, DOI: https://doi.org/10.1002/rob.21809. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21809. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21809.

[165] G. Pandey, S. Savarese, J. R. McBride and R. M. Eustice, „Visually bootstrapped generalized ICP,“ in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2660–2667, DOI: 10.1109/ICRA.2011.5980322.

[166] J. Zhang and S. Singh, „Laser–visual–inertial odometry and mapping with high robustness and low drift,“ *Journal of field robotics*, vol. 35, no. 8, pp. 1242–1264, 2018, DOI: https://doi.org/10.1002/rob.21809.

[167] Z. Zhu, S. Yang, H. Dai and F. Li, „Loop detection and correction of 3d laser-based slam with visual information,“ in *Proceedings of the 31st International Conference on Computer Animation and Social Agents*, 2018, pp. 53–58, DOI: https://doi.org/10.1145/3205326.3205357.

[168] Y. Li, T. Li and H. Liu, „Recent advances in feature selection and its applications,“ *Knowledge and Information Systems*, vol. 53, pp. 551–577, 2017, DOI: https://doi.org/10.1007/s10115-017-1059-8.

[169] X. Xu, L. Zhang, J. Yang, C. Cao, W. Wang, Y. Ran, Z. Tan and M. Luo, „A Review of Multi-Sensor Fusion SLAM Systems Based on 3D LIDAR,“ *Remote Sensing*, vol. 14, no. 12, 2022, DOI: 10.3390/rs14122835.

[170] W. Shao, S. Vijayarangan, C. Li and G. Kantor, „Stereo Visual Inertial LiDAR Simultaneous Localization and Mapping,“ in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 370–377, DOI: 10.1109/IROS40897.2019.8968012.

[171] X. Zuo, P. Geneva, W. Lee, Y. Liu and G. Huang, „LIC-Fusion: LiDAR-Inertial-Camera Odometry,“ in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5848–5854, DOI: 10.1109/IROS40897.2019.8967746.

[172] X. Zuo, Y. Yang, P. Geneva, J. Lv, Y. Liu, G. Huang and M. Pollefeys, „LIC-Fusion 2.0: LiDAR-Inertial-Camera Odometry with Sliding-Window Plane-Feature Tracking,“ in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5112–5119, DOI: 10.1109/IROS45743.2020.9340704.

[173] T. Shan, B. Englot, C. Ratti and D. Rus, „LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping,“ in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5692–5698, DOI: 10.1109/ICRA48506.2021.9561996.

[174] S. Zhao, H. Zhang, P. Wang, L. Nogueira and S. Scherer, „Super Odometry: IMU-centric LiDAR-Visual-Inertial Estimator for Challenging Environments,“ in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 8729–8736, DOI: 10.1109/IROS51168.2021.9635862.

[175] J. Lin, C. Zheng, W. Xu and F. Zhang, „R ^{2} LIVE: A Robust, Real-Time, LiDAR-Inertial-Visual Tightly-Coupled State Estimator and Mapping,“ *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7469–7476, 2021, DOI: 10.1109/LRA.2021.3095515.

[176] J. Lin and F. Zhang, „R3LIVE: A Robust, Real-time, RGB-colored, LiDAR-Inertial-Visual tightly-coupled state Estimation and mapping package,“ in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10672–10678, DOI: 10.1109/ICRA46639.2022.9811935.

[177] D. Wisth, M. Camurri, S. Das and M. Fallon, „Unified Multi-Modal Landmark Tracking for Tightly Coupled Lidar-Visual-Inertial Odometry,“ *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1004–1011, 2021, DOI: 10.1109/LRA.2021.3056380.

[178] C. Zheng, Q. Zhu, W. Xu, X. Liu, Q. Guo and F. Zhang, „FAST-LIVO: Fast and Tightly-coupled Sparse-Direct LiDAR-Inertial-Visual Odometry,“ *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4003–4009, 2022. Available: https://api.semanticscholar.org/CorpusID:247218345.

[179] C.-C. Chou and C.-F. Chou, „Efficient and Accurate Tightly-Coupled Visual-Lidar SLAM,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14509–14523, 2022, DOI: 10.1109/TITS.2021.3130089.

[180] X. Lang, C. Chen, K. Tang, Y. Ma, J. Lv, Y. Liu and X. Zuo, „Coco-LIC: Continuous-Time Tightly-Coupled LiDAR-Inertial-Camera Odometry using Non-Uniform B-spline,“ *IEEE Robotics and Automation Letters*, pp. 1–8, 2023, DOI: 10.1109/LRA.2023.3315542.

[181] Z. Yuan, Q. Wang, K. Cheng, T. Hao and X. Yang, „SDV-LOAM: Semi-Direct Visual–LiDAR Odometry and Mapping,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 11203–11220, 2023, DOI: 10.1109/TPAMI.2023.3262817.

[182] W. Chen, C. Zhou, G. Shang, X. Wang, Z. Li, C. Xu and K. Hu, „SLAM Overview: From Single Sensor to Heterogeneous Fusion,“ *Remote Sensing*, vol. 14, no. 23, p. 6033, 2022, DOI: https://doi.org/10.3390/rs14236033.

[183] J. Zhu, H. Li and T. Zhang, „Camera, LiDAR, and IMU Based Multi-Sensor Fusion SLAM: A Survey,“ *Tsinghua Science and Technology*, vol. 29, no. 2, pp. 415–429, 2024, DOI: 10.26599/TST.2023.9010010.

[184] A. Khoche, M. K. Wozniak, D. Duberg and P. Jensfelt, „Semantic 3D Grid Maps for Autonomous Driving,“ in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 2681–2688, DOI: https://doi.org/10.1109/ITSC55140.2022.9922537.

[185] H. G. Seif and X. Hu, „Autonomous driving in the iCity—HD maps as a key challenge of the automotive industry,“ *Engineering*, vol. 2, no. 2, pp. 159–162, 2016, DOI: https://doi.org/10.1016/J.ENG.2016.02.010.

[186] Z. Bao, S. Hossain, H. Lang and X. Lin, „A review of high-definition map creation methods for autonomous driving,“ *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106125, 2023, DOI: https://doi.org/10.1016/j.engappai.2023.106125.

[187] L. Zheng, B. Li, B. Yang, H. Song and Z. Lu, „Lane-Level Road Network Generation Techniques for Lane-Level Maps of Autonomous Vehicles: A Survey,“ *Sustainability*, vol. 11, no. 16, 2019, DOI: 10.3390/su11164511. Available: https://www.mdpi.com/2071-1050/11/16/4511.

[188] J. Jeong, J. Y. Yoon, H. Lee, H. Darweesh and W. Sung, „Tutorial on High-Definition Map Generation for Automated Driving in Urban Environments,“ *Sensors*, vol. 22, no. 18, 2022, DOI: https://doi.org/10.3390/s22187056. Available: https://www.mdpi.com/1424-8220/22/18/7056.

[189] R. Liu, J. Wang and B. Zhang, „High definition map for automated driving: Overview and analysis,“ *The Journal of Navigation*, vol. 73, no. 2, pp. 324–341, 2020, DOI: https://doi.org/10.1017/S0373463319000638.

[190] B. Ebrahimi Soorchaei, M. Razzaghpour, R. Valiente, A. Raftari and Y. P. Fallah, „High-Definition Map Representation Techniques for Automated Vehicles,“ *Electronics*, vol. 11, no. 20, 2022, DOI: 10.3390/electronics11203374. Available: https://www.mdpi.com/2079-9292/11/20/3374.

[191] Automotive Edge Computing Consortium. „*Operational Behavior of a High Definition Map Application*,“ 2020. Available: https://aecc.org/wp-content/uploads/2020/07/Operational_Behavior_of_a_High_Definition_Map_Application.pdf.

[192] HERE. „HERE HD Live Map,“ 2023. [Online]. Available: https://www.here.com/platform/HD-live-map [visited on 03/13/2023].

[193] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt and M. Mayr, „Lanelet2: A high-definition map framework for the future of automated driving,“ in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 1672–1679, DOI: 10.1109/ITSC.2018.8569929.

[194] M. Haklay and P. Weber, „OpenStreetMap: User-Generated Street Maps,“ *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008, DOI: https://doi.org/10.1109/MPRV.2008.80.

[195] ASAM. „ASAM OpenDRIVE,“ 2023. [Online]. Available: https://www.asam.net/standards/detail/opendrive/ [visited on 09/13/2023].

[196] M. Siopi, D. Ellinoudis, I. Pratikakis and A. Amanatiadis, „A study on high definition maps' standards and specifications for autonomous vehicles,“ in *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2023, pp. 1–7, DOI: 10.1109/COINS57856.2023.10189236.

[197] T. Yu, H. Huang, N. Jiang and T. D. Acharya, „Study on Relative Accuracy and Verification Method of High-Definition Maps for Autonomous Driving,“ *ISPRS International Journal of Geo-Information*, vol. 10, no. 11, 2021, DOI: 10.3390/ijgi10110761. Available: https://www.mdpi.com/2220-9964/10/11/761.

[198] C. Maps. „Civil Maps | HD maps and localization,“ 2023. [Online]. Available: https://civilmaps.com/ [visited on 11/14/2023].

[199] J. M. Lógó, N. Krausz, V. Potó and A. Barsi, „QUALITY ASPECTS OF HIGH-DEFINITION MAPS,“ *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B4-2021, pp. 389–394, 2021, DOI: 10.5194/isprs-archives-XLIII-B4-2021-389-2021. Available: https://isprs-archives.copernicus.org/articles/XLIII-B4-2021/389/2021/.

[200] D. Pannen, M. Liebner, W. Hempel and W. Burgard, „How to Keep HD Maps for Automated Driving Up To Date,“ in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2288–2294, DOI: 10.1109/ICRA40945.2020.9197419.

[201] A. Diaz-Diaz, M. Ocaña, Á. Llamazares, C. Gómez-Huélamo, P. Revenga and L. M. Bergasa, „HD maps: Exploiting OpenDRIVE potential for Path Planning and Map Monitoring,“ in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1211–1217, DOI: 10.1109/IV51971.2022.9827297.

[202] K. Massow, B. Kwella, N. Pfeifer, F. Häusler, J. Pontow, I. Radusch, J. Hipp, F. Dölitzscher and M. Haueis, „Deriving HD maps for highly automated driving from vehicular probe data,“ in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1745–1752, DOI: 10.1109/ITSC.2016.7795794.

[203] K. Kim, S. Cho and W. Chung, „HD Map Update for Autonomous Driving With Crowdsourced Data,“ *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1895–1901, 2021, DOI: 10.1109/LRA.2021.3060406.

[204] P. Fischer, S. M. Azimi, R. Roschlaub and T. Krauß, „Towards HD Maps from Aerial Imagery: Robust Lane Marking Segmentation Using Country-Scale Imagery,“ *ISPRS International Journal of Geo-Information*, vol. 7, no. 12, 2018, DOI: 10.3390/ijgi7120458. Available: https://www.mdpi.com/2220-9964/7/12/458.

[205] K. Wong, Y. Gu and S. Kamijo, „Mapping for Autonomous Driving: Opportunities and Challenges,“ *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 1, pp. 91–106, 2021, DOI: 10.1109/MITS.2020.3014152.

[206] S. Shi, C. Hu, D. Wang, Y. Zhu and Z. Han, „Federated HD Map Updating Through Overlapping Coalition Formation Game,“ *IEEE Transactions on Mobile Computing*, pp. 1–14, 2023, DOI: 10.1109/TMC.2023.3241090.

[207] F. Zhang, W. Shi, M. Chen, W. Huang and X. Liu, „Open HD map service model: an interoperable high-Definition map data model for autonomous driving,“ *International Journal of Digital Earth*, vol. 16, no. 1, pp. 2089–2110, 2023, DOI: 10.1080/17538947.2023.2220615.

[208] T. Bock, „Unsettled issues on HD mapping technology for autonomous driving and ADAS,“ SAE Technical Paper, 2021.

[209] T. Zheng, C. Chen, J. Yuan, B. Li and K. Ren, „PointCloud Saliency Maps,“ in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1598–1606, DOI: 10.1109/ICCV.2019.00168.

[210] H. Wang, H. Luo, C. Wen, J. Cheng, P. Li, Y. Chen, C. Wang and J. Li, „Road Boundaries Detection Based on Local Normal Saliency From Mobile Laser Scanning Data,“ *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 10, pp. 2085–2089, 2015, DOI: 10.1109/LGRS.2015.2449074.

[211] X. Mi, B. Yang, Z. Dong, C. Chen and J. Gu, „Automated 3D Road Boundary Extraction and Vectorization Using MLS Point Clouds,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5287–5297, 2022, DOI: 10.1109/TITS.2021.3052882.

[212] D. Zai, J. Li, Y. Guo, M. Cheng, Y. Lin, H. Luo and C. Wang, „3-D Road Boundary Extraction From Mobile Laser Scanning Data via Supervoxels and Graph Cuts,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 802–813, 2018, DOI: 10.1109/TITS.2017.2701403.

[213] S. Xu, R. Wang and H. Zheng, „Road Curb Extraction From Mobile LiDAR Point Clouds,“ *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 996–1009, 2017, DOI: 10.1109/TGRS.2016.2617819.

[214] H. Guan, J. Li, Y. Yu, C. Wang, M. Chapman and B. Yang, „Using mobile laser scanning data for automated extraction of road markings,“ *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 93–107, 2014, DOI: https://doi.org/10.1016/j.isprsjprs.2013.11.005.

[215] L. Ma, Y. Li, J. Li, Z. Zhong and M. A. Chapman, „Generation of Horizontally Curved Driving Lines in HD Maps Using Mobile Laser Scanning Point Clouds,“ *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 5, pp. 1572–1586, 2019, DOI: 10.1109/JSTARS.2019.2904514.

[216] C. Ye, H. Zhao, L. Ma, H. Jiang, H. Li, R. Wang, M. A. Chapman, J. M. Junior and J. Li, „Robust Lane Extraction From MLS Point Clouds Towards HD Maps Especially in Curve Road,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1505–1518, 2022, DOI: 10.1109/TITS.2020.3028033.

[217] B. Yang, L. Fang and J. Li, „Semi-automated extraction and delineation of 3D roads of street scene from mobile laser scanning point clouds,“ *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 79, pp. 80–93, 2013, DOI: https://doi.org/10.1016/j.isprsjprs.2013.01.016.

[218] M. Yadav, A. K. Singh and B. Lohani, „Extraction of road surface from mobile LiDAR data of complex road environment,“ *International Journal of Remote Sensing*, vol. 38, no. 16, pp. 4655–4682, 2017, DOI: 10.1080/01431161.2017.1320451.

[219] M. Yadav, B. Lohani and A. Singh, „Road surface detection from mobile lidar data,“ *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, pp. 95–101, 2018, DOI: https://doi.org/10.5194/isprs-annals-IV-5-95-2018.

[220] M. Yadav and A. K. Singh, „Rural road surface extraction using mobile LiDAR point cloud data,“ *Journal of the Indian Society of Remote Sensing*, vol. 46, pp. 531–538, 2018, DOI: https://doi.org/10.1007/s12524-017-0732-4.

[221] Z. Hui, Y. Hu, S. Jin and Y. Z. Yevenyo, „Road centerline extraction from airborne LiDAR point cloud based on hierarchical fusion and optimization,“ *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 118, pp. 22–36, 2016, DOI: https://doi.org/10.1016/j.isprsjprs.2016.04.003.

[222] M. Cheng, H. Zhang, C. Wang and J. Li, „Extraction and Classification of Road Markings Using Mobile Laser Scanning Point Clouds,“ *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 3, pp. 1182–1196, 2017, DOI: 10.1109/JSTARS.2016.2606507.

[223] A. Hata and D. Wolf, „Road marking detection using LIDAR reflective intensity data and its application to vehicle localization,“ in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 584–589, DOI: 10.1109/ITSC.2014.6957753.

[224] P. Kumar, C. P. McElhinney, P. Lewis and T. McCarthy, „Automated road markings extraction from mobile laser scanning data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 32, pp. 125–137, 2014, DOI: https://doi.org/10.1016/j.jag.2014.03.023.

[225] J. Jung, E. Che, M. J. Olsen and C. Parrish, „Efficient and robust lane marking extraction from mobile lidar point clouds," *ISPRS journal of photogrammetry and remote sensing*, vol. 147, pp. 1–18, 2019, DOI: https://doi.org/10.1016/j.isprsjprs.2018.11.012.

[226] B. Riveiro, H. González-Jorge, J. Martínez-Sánchez, L. Díaz-Vilariño and P. Arias, „Automatic detection of zebra crossings from mobile LiDAR data," *Optics & Laser Technology*, vol. 70, pp. 63–70, 2015, DOI: https://dx.doi.org/10.1016/j.optlastec.2015.01.011.

[227] Y. Yu, J. Li, H. Guan, F. Jia and C. Wang, „Learning Hierarchical Features for Automated Extraction of Road Markings From 3-D Mobile LiDAR Point Clouds," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 2, pp. 709–726, 2015, DOI: 10.1109/JSTARS.2014.2347276.

[228] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, „PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85, DOI: 10.1109/CVPR.2017.16.

[229] C. R. Qi, L. Yi, H. Su and L. J. Guibas, „PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems*, 2017. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[230] A. Milioto, I. Vizzo, J. Behley and C. Stachniss, „RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4213–4220, DOI: 10.1109/IROS40897.2019.8967762.

[231] Y. Wang, T. Shi, P. Yun, L. Tai and M. Liu, „Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *arXiv preprint arXiv:1807.06288*, 2018.

[232] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein and J. M. Solomon, „Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019, DOI: https://doi.org/10.1145/3326362.

[233] M. Rochan, S. Aich, E. R. Corral-Soto, A. Nabatchian and B. Liu, „Unsupervised Domain Adaptation in LiDAR Semantic Segmentation with Self-Supervision and Gated Adapters," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2649–2655, DOI: 10.1109/ICRA46639.2022.9811654.

[234] P. V. Hough. „*Method and means for recognizing complex patterns*," US Patent 3,069,654. 1962.

[235] D. C. Andrade, F. Bueno, F. R. Franco, R. A. Silva, J. H. Z. Neme, E. Margraf, W. T. Omoto, F. A. Farinelli, A. M. Tusset, S. Okida, M. M. D. Santos, A. Ventura, S. Carvalho and R. d. S. Amaral, „A Novel Strategy for Road Lane Detection and Tracking Based on a Vehicle's Forward Monocular Camera," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 4, pp. 1497–1507, 2019, DOI: 10.1109/TITS.2018.2856361.

[236] Z. Zhang, X. Zhang, S. Yang and J. Yang, „Research on Pavement Marking Recognition and Extraction Method," in *2021 6th International Conference on Image, Vision and Computing (ICIVC)*, 2021, pp. 100–105, DOI: 10.1109/ICIVC52351.2021.9526998.

[237] Z. Qin, H. Wang and X. Li, „Ultra fast structure-aware deep lane detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, 2020, pp. 276–291, DOI: https://doi.org/10.1007/978-3-030-58586-0_17.

[238] S. Yoo, H. Seok Lee, H. Myeong, S. Yun, H. Park, J. Cho and D. Hoon Kim, „End-to-End Lane Marker Detection via Row-wise Classification,“ in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 4335–4343, DOI: 10.1109/CVPRW50498.2020.00511.

[239] J. Shi, J. Wang and J. Li, „Fast Lane Detection Flexibly Adapting to Road Structure Information,“ in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 6605–6609, DOI: 10.23919/CCC55666.2022.9901803.

[240] Z. Chen, Q. Liu and C. Lian, „PointLaneNet: Efficient end-to-end CNNs for Accurate Real-Time Lane Detection,“ in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2563–2568, DOI: 10.1109/IVS.2019.8813778.

[241] N. Garnett, R. Cohen, T. Pe'er, R. Lahav and D. Levi, „3D-LaneNet: End-to-End 3D Multiple Lane Detection,“ in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2921–2930, DOI: 10.1109/ICCV.2019.00301.

[242] H. Xu, S. Wang, X. Cai, W. Zhang, X. Liang and Z. Li, „Curvelane-nas: Unifying lane-sensitive architecture search and adaptive point blending,“ in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, 2020, pp. 689–704, DOI: https://doi.org/10.1007/978-3-030-58555-6_41.

[243] L. Tabelini, R. Berriel, T. M. Paixão, C. Badue, A. F. De Souza and T. Oliveira-Santos, „Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection,“ in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 294–302, DOI: 10.1109/CVPR46437.2021.00036.

[244] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, „An image is worth 16x16 words: Transformers for image recognition at scale,“ *arXiv preprint arXiv:2010.11929*, 2020.

[245] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez and P. Luo, „SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers,“ in *Advances in Neural Information Processing Systems*, 2021, pp. 12077–12090. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/64f1f27bf1b4ec22924fd0acb550c235-Paper.pdf.

[246] Y. Mo, Y. Wu, X. Yang, F. Liu and Y. Liao, „Review the state-of-the-art technologies of semantic segmentation based on deep learning,“ *Neurocomputing*, vol. 493, pp. 626–646, 2022, DOI: https://doi.org/10.1016/j.neucom.2022.01.005.

[247] H. Thisanke, C. Deshan, K. Chamith, S. Seneviratne, R. Vidanaarachchi and D. Herath, „Semantic segmentation using Vision Transformers: A survey,“ *Engineering Applications of Artificial Intelligence*, vol. 126, p. 106669, 2023, DOI: https://doi.org/10.1016/j.engappai.2023.106669.

[248] S. Gu, T. Lu, Y. Zhang, J. M. Alvarez, J. Yang and H. Kong, „3-D LiDAR + Monocular Camera: An Inverse-Depth-Induced Fusion Framework for Urban Road Detection,“ *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 351–360, 2018, DOI: 10.1109/TIV.2018.2843170.

[249] F. Yang, J. Yang, Z. Jin and H. Wang, „A Fusion Model for Road Detection based on Deep Learning and Fully Connected CRF,“ in *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, 2018, pp. 29–36, DOI: 10.1109/SYSOSE.2018.8428696.

[250] X. Lv, Z. Liu, J. Xin and N. Zheng, „A Novel Approach for Detecting Road Based on Two-Stream Fusion Fully Convolutional Network,“ in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1464–1469, DOI: 10.1109/IVS.2018.8500551.

[251] L. Caltagirone, M. Bellone, L. Svensson and M. Wahde, „LIDAR–camera fusion for road detection using fully convolutional neural networks,“ *Robotics and Autonomous Systems*, vol. 111, pp. 125–131, 2019, DOI: https://doi.org/10.1016/j.robot.2018.11.002. Available: https://www.sciencedirect.com/science/article/pii/S0921889018300496.

[252] H. Guan, X. Lei, Y. Yu, H. Zhao, D. Peng, J. M. Junior and J. Li, „Road marking extraction in UAV imagery using attentive capsule feature pyramid network,“ *International Journal of Applied Earth Observation and Geoinformation*, vol. 107, p. 102677, 2022, DOI: https://doi.org/10.1016/j.jag.2022.102677.

[253] P. Shamsolmoali, M. Zareapoor, H. Zhou, R. Wang and J. Yang, „Road Segmentation for Remote Sensing Images Using Adversarial Spatial Pyramid Networks,“ *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 6, pp. 4673–4688, 2021, DOI: 10.1109/TGRS.2020.3016086.

[254] Y. Yu, Y. Li, C. Liu, J. Wang, C. Yu, X. Jiang, L. Wang, Z. Liu and Y. Zhang, „MarkCapsNet: Road Marking Extraction From Aerial Images Using Self-Attention-Guided Capsule Network,“ *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022, DOI: 10.1109/LGRS.2021.3124575.

[255] S. M. Azimi, P. Fischer, M. Körner and P. Reinartz, „Aerial LaneNet: Lane-Marking Semantic Segmentation in Aerial Imagery Using Wavelet-Enhanced Cost-Sensitive Symmetric Fully Convolutional Neural Networks,“ *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 5, pp. 2920–2938, 2019, DOI: 10.1109/TGRS.2018.2878510.

[256] Y. Liu, J. Yao, X. Lu, M. Xia, X. Wang and Y. Liu, „RoadNet: Learning to Comprehensively Analyze Road Networks in Complex Urban Scenes From High-Resolution Remotely Sensed Images,“ *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 4, pp. 2043–2056, 2019, DOI: 10.1109/TGRS.2018.2870871.

[257] K. Yang, W. Cui, S. Shi, Y. Liu, Y. Li and M. Ge, „Semi-Automatic Method of Extracting Road Networks from High-Resolution Remote-Sensing Images,“ *Applied Sciences*, vol. 12, no. 9, 2022, DOI: 10.3390/app12094705. Available: https://www.mdpi.com/2076-3417/12/9/4705.

[258] G.-P. Gwon, W.-S. Hur, S.-W. Kim and S.-W. Seo, „Generation of a Precise and Efficient Lane-Level Road Map for Intelligent Vehicle Systems,“ *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 4517–4533, 2017, DOI: 10.1109/TVT.2016.2535210.

[259] Z. Yu, H. Zhu, L. Lin, H. Liang, B. Yu and W. Huang, „Laser Data Based Automatic Generation of Lane-Level Road Map for Intelligent Vehicles,“ *arXiv preprint arXiv:2101.05066*, 2020.

[260] S. He and H. Balakrishnan, „Lane-Level Street Map Extraction from Aerial Imagery,“ in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 1496–1505, DOI: 10.1109/WACV51458.2022.00156.

[261] Q. Li, Y. Wang, Y. Wang and H. Zhao, „HDMapNet: An Online HD Map Construction and Evaluation Framework,“ in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4628–4634, DOI: 10.1109/ICRA46639.2022.9812383.

[262] Y. Liu, T. Yuan, Y. Wang, Y. Wang and H. Zhao, „VectorMapNet: End-to-end Vectorized HD Map Learning,“ in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 22352–22369.

[263] B. Liao, S. Chen, X. Wang, T. Cheng, Q. Zhang, W. Liu and C. Huang, „Maptr: Structured modeling and learning for online vectorized hd map construction,“ *arXiv preprint arXiv:2208.14437*, 2022.

[264] B. Liao, S. Chen, Y. Zhang, B. Jiang, Q. Zhang, W. Liu, C. Huang and X. Wang, „MapTRv2: An End-to-End Framework for Online Vectorized HD Map Construction,“ *arXiv preprint arXiv:2308.05736*, 2023.

[265] H. Dong, X. Zhang, J. Xu, R. Ai, W. Gu, H. Lu, J. Kannala and X. Chen. „*SuperFusion: Multilevel LiDAR-Camera Fusion for Long-Range HD Map Generation*,“ 2023. arXiv: 2211.15656 [cs.CV].

[266] J. Shin, F. Rameau, H. Jeong and D. Kum, „Instagram: Instance-level graph modeling for vectorized hd map learning,“ *arXiv preprint arXiv:2301.04470*, 2023.

[267] W. Ding, L. Qiao, X. Qiu and C. Zhang, „PivotNet: Vectorized Pivot Learning for End-to-end HD Map Construction,“ *arXiv preprint arXiv:2308.16477*, 2023.

[268] S. Wang, W. Li, W. Liu, X. Liu and J. Zhu, „LiDAR2Map: In Defense of LiDAR-Based Semantic Map Construction Using Online Camera Distillation,“ in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 5186–5195, DOI: 10.1109/CVPR52729.2023.00502.

[269] D. Paz, H. Zhang, Q. Li, H. Xiang and H. I. Christensen, „Probabilistic Semantic Mapping for Urban Autonomous Driving Applications,“ in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 2059–2064, DOI: 10.1109/IROS45743.2020.9341738.

[270] H. Zhang, S. Venkatramani, D. Paz, Q. Li, H. Xiang and H. I. Christensen, „Probabilistic Semantic Mapping for Autonomous Driving in Urban Environments,“ *Sensors*, vol. 23, no. 14, 2023, DOI: 10.3390/s23146504. Available: https://www.mdpi.com/1424-8220/23/14/6504.

[271] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. L. Rus and S. Han, „BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird’s-Eye View Representation,“ in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2774–2781, DOI: 10.1109/ICRA48891.2023.10160968.

[272] T. Liang, H. Xie, K. Yu, Z. Xia, Z. Lin, Y. Wang, T. Tang, B. Wang and Z. Tang, „BEVFusion: A Simple and Robust LiDAR-Camera Fusion Framework,“ in *Advances in Neural Information Processing Systems*, 2022, pp. 10421–10434. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/43d2b7fbee8431f7cef0d0afed51c691-Paper-Conference.pdf.

[273] S. Li, K. Yang, H. Shi, J. Zhang, J. Lin, Z. Teng and Z. Li, „Bi-Mapper: Holistic BEV Semantic Mapping for Autonomous Driving,“ *arXiv preprint arXiv:2305.04205*, 2023.

[274] A. W. Harley, Z. Fang, J. Li, R. Ambrus and K. Fragkiadaki, „Simple-BEV: What Really Matters for Multi-Sensor BEV Perception?,“ in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2759–2765, DOI: 10.1109/ICRA48891.2023.10160831.

[275] N. Gosala, K. Petek, P. L. J. Drews-Jr, W. Burgard and A. Valada, „SkyEye: Self-Supervised Bird’s-Eye-View Semantic Mapping Using Monocular Frontal View Images,“ in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 14901–14910, DOI: 10.1109/CVPR52729.2023.01431.

[276] C. Sima, W. Tong, T. Wang, L. Chen, S. Wu, H. Deng, Y. Gu, L. Lu, P. Luo, D. Lin and H. Li, „Scene as Occupancy,“ 2023. arXiv: 2306.02851 [cs.CV].

[277] X. Tian, T. Jiang, L. Yun, Y. Wang, Y. Wang and H. Zhao, „Occ3D: A Large-Scale 3D Occupancy Prediction Benchmark for Autonomous Driving,“ *arXiv preprint arXiv:2304.14365*, 2023.

[278] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, J. Zhou and J. Lu, „SurroundOcc: Multi-camera 3D Occupancy Prediction for Autonomous Driving,“ in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 21729–21740.

[279] L. Nunes, L. Wiesmann, R. Marcuzzi, X. Chen, J. Behley and C. Stachniss, „Temporal Consistent 3D LiDAR Representation Learning for Semantic Perception in Autonomous Driving,“ in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 5217–5228, DOI: 10.1109/CVPR52729.2023.00505.

[280] P. Li, T. Qin and a. Shen, „Stereo Vision-based Semantic 3D Object and Ego-motion Tracking for Autonomous Driving,“ in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[281] A. Ando, S. Gidaris, A. Bursuc, G. Puy, A. Boulch and R. Marlet, „RangeViT: Towards Vision Transformers for 3D Semantic Segmentation in Autonomous Driving,“ in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 5240–5250, DOI: 10.1109/CVPR52729.2023.00507.

[282] D. Pannen, M. Liebner and W. Burgard, „HD Map Change Detection with a Boosted Particle Filter,“ in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2561–2567, DOI: 10.1109/ICRA.2019.8794329.

[283] B. Wijaya, K. Jiang, M. Yang, T. Wen, X. Tang, D. Yang, Y. Ma and R. Albert, „CrowdRep: A Blockchain-based Reputation System for Crowdsourced HD Map Update,“ in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 3050–3057, DOI: 10.1109/ITSC55140.2022.9922458.

[284] C. Kim, S. Cho, M. Sunwoo, P. Resende, B. Bradaï and K. Jo, „Updating Point Cloud Layer of High Definition (HD) Map Based on Crowd-Sourcing of Multiple Vehicles Installed LiDAR,“ *IEEE Access*, vol. 9, pp. 8028–8046, 2021, DOI: 10.1109/ACCESS.2021.3049482.

[285] K. Jo, C. Kim and M. Sunwoo, „Simultaneous Localization and Map Change Update for the High Definition Map-Based Autonomous Driving Car,“ *Sensors*, vol. 18, no. 9, 2018, DOI: 10.3390/s18093145. Available: https://www.mdpi.com/1424-8220/18/9/3145.

[286] J. S. Berrio, S. Worrall, M. Shan and E. Nebot, „Long-Term Map Maintenance Pipeline for Autonomous Vehicles,“ *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10427–10440, 2022, DOI: 10.1109/TITS.2021.3094485.

[287] G. Elghazaly, R. Frank, S. Harvey and S. Safko, „High-Definition Maps: Comprehensive Survey, Challenges, and Future Perspectives,“ *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 527–550, 2023, DOI: 10.1109/OJITS.2023.3295502.

[288] A. Boubakri, S. M. Gammar, M. B. Brahim and F. Filali, „High definition map update for autonomous and connected vehicles: A survey,“ in *2022 International Wireless Communications and Mobile Computing (IWCMC)*, 2022, pp. 1148–1153, DOI: 10.1109/IWCMC55113.2022.9825276.

[289] D. Shin, K.-m. Park and M. Park, „High Definition Map-Based Localization Using ADAS Environment Sensors for Application to Automated Driving Vehicles,“ *Applied Sciences*, vol. 10, no. 14, 2020, DOI: 10.3390/app10144924. Available: https://www.mdpi.com/2076-3417/10/14/4924.

[290] D. Rozenberszki and A. L. Majdik, „LOL: Lidar-only Odometry and Localization in 3D point cloud maps,“ in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4379–4385, DOI: 10.1109/ICRA40945.2020.9197450.

[291] B. Nagy and C. Benedek, „Real-time point cloud alignment for vehicle localization in a high resolution 3D map,“ in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[292] C. Zhang, L. Liu, Z. Xue, K. Guo, K. Yang, R. Cai and Z. Li, „Robust LiDAR Localization on an HD Vector Map without a Separate Localization Layer,“ in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5536–5543, DOI: 10.1109/IROS51168.2021.9636227.

[293] Z. Xiao, D. Yang, T. Wen, K. Jiang and R. Yan, „Monocular Localization with Vector HD Map (MLVHM): A Low-Cost Method for Commercial IVs,“ *Sensors*, vol. 20, no. 7, 2020, DOI: 10.3390/s20071870. Available: https://www.mdpi.com/1424-8220/20/7/1870.

[294] W.-C. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky and R. Urtasun, „Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization,“ in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5304–5311, DOI: 10.1109/IROS40897.2019.8968122.

[295] X. Xiong, Y. Liu, T. Yuan, Y. Wang, Y. Wang and H. Zhao, „Neural Map Prior for Autonomous Driving,“ in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 17535–17544, DOI: 10.1109/CVPR52729.2023.01682.

[296] Z. Luo, L. Gao, H. Xiang and J. Li, „Road object detection for HD map: Full-element survey, analysis and perspectives,“ *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 197, pp. 122–144, 2023, DOI: https://doi.org/10.1016/j.isprsjprs.2023.01.009.

[297] B. Yang, M. Liang and R. Urtasun, „HDNET: Exploiting HD Maps for 3D Object Detection,“ in *Proceedings of The 2nd Conference on Robot Learning*, 2018, pp. 146–155. Available: https://proceedings.mlr.press/v87/yang18b.html.

[298] B. Ravi Kiran, L. Roldao, B. Irastorza, R. Verastegui, S. Suss, S. Yogamani, V. Talpaert, A. Lepoutre and G. Trehard, „Real-time dynamic object detection for autonomous driving using prior 3d-maps,“ in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[299] Y. Huang, J. Zhou, X. Li, Z. Dong, J. Xiao, S. Wang and H. Zhang, „MENet: Map-enhanced 3D object detection in bird's-eye view for LiDAR point clouds,“ *International Journal of Applied Earth Observation and Geoinformation*, vol. 120, p. 103337, 2023, DOI: https://doi.org/10.1016/j.jag.2023.103337.

[300] J.-R. Xue, J.-W. Fang and P. Zhang, „A survey of scene understanding by event reasoning in autonomous driving,“ *International Journal of Automation and Computing*, vol. 15, no. 3, pp. 249–266, 2018, DOI: https://doi.org/10.1007/s11633-018-1126-y.

[301] W.-C. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky and R. Urtasun, „Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization,“ in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5304–5311, DOI: 10.1109/IROS40897.2019.8968122.

[302] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov and B. Sapp, „MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction,“ in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7814–7821, DOI: 10.1109/ICRA46639.2022.9812107.

[303] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat and B. Sapp, „Wayformer: Motion Forecasting via Simple & Efficient Attention Networks,“ in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2980–2987, DOI: 10.1109/ICRA48891.2023.10160609.

[304] Z. Wang, J. Guo, Z. Hu, H. Zhang, J. Zhang and J. Pu, „Lane Transformer: A High-Efficiency Trajectory Prediction Model,“ *IEEE Open Journal of Intelligent Transportation Systems*, vol. 4, pp. 2–13, 2023, DOI: 10.1109/OJITS.2023.3233952.

[305] Z. Jian, S. Zhang, S. Chen, X. Lv and N. Zheng, „High-Definition Map Combined Local Motion Planning and Obstacle Avoidance for Autonomous Driving,“ in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2180–2186, DOI: 10.1109/IVS.2019.8814229.

[306] X. Guo, Y. Cao, J. Zhou, Y. Huang and B. Li, „HDM-RRT: A Fast HD-Map-Guided Motion Planning Algorithm for Autonomous Driving in the Campus Environment,“ *Remote Sensing*, vol. 15, no. 2, 2023, DOI: 10.3390/rs15020487. Available: https://www.mdpi.com/2072-4292/15/2/487.

[307] A. Wischnewski, M. Geisslinger, J. Betz, T. Betz, F. Fent, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, F. Nobis, L. Ögretmen, M. Rowold, **F. Sauerbeck**, T. Stahl, R. Trauth, M. Lienkamp and B. Lohmann, „Indy autonomous challenge-autonomous race cars at the handling limits,“ in *12th International Munich Chassis Symposium 2021: chassis. tech plus*, 2022, pp. 163–182.

[308] „Indy Lights - Dallara IL15,“ [Online]. Available: https://www.dallara.it/en/racing/indy%5C%20lights.

[309] F. Massa, L. Bonamini, A. Settimi, L. Pallottino and D. Caporale, „Lidar-based gnss denied localization for autonomous racing cars,“ *Sensors*, vol. 20, no. 14, p. 3992, 2020, DOI: https://doi.org/10.3390/s20143992.

[310] M. Schratter, J. Zubaca, K. Mautner-Lassnig, T. Renzler, M. Kirchengast, S. Loigge, M. Stolz and D. Watzenig, „LiDAR-based mapping and Localization for autonomous racing,“ in *Proc. Int. Conf. Robot. Autom.(ICRA) Workshop Opportunities Challenges Auton. Racing*, 2021, pp. 1–6.

[311] T. Stahl, A. Wischnewski, J. Betz and M. Lienkamp, „ROS-based localization of a race vehicle at high-speed using LIDAR,“ in *E3S Web of Conferences*, 2019, p. 04002.

[312] R. B. Rusu and S. Cousins, „3D is here: Point Cloud Library (PCL),“ in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4, DOI: 10.1109/ICRA.2011.5980567.

[313] K. Koide. „ndt_omp: Mulit-threaded and SSE friendly NDT algorithm,“ 2023. [Online]. Available: https://github.com/koide3/ndt_omp [visited on 11/23/2023].

[314] K. Koide, M. Yokozuka, S. Oishi and A. Banno, „Voxelized GICP for Fast and Accurate 3D Point Cloud Registration,“ in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11054–11059, DOI: 10.1109/ICRA48506.2021.9560835.

[315] D. Kulmer, „Combined Map Based Localization and Object Detection for Autonomous Race Cars,“ Master's Thesis, Technische Universität München, München, 2022.

[316] K. Koide, J. Miura and E. Menegatti, „A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement,“ *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, 2019, DOI: https://doi.org/10.1177/1729881419841.

[317] K. Koide, J. Miura, M. Yokozuka, S. Oishi and A. Banno, „Interactive 3D Graph SLAM for Map Correction,“ *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 40–47, 2021, DOI: 10.1109/LRA.2020.3028828.

[318] M. Werling, J. Ziegler, S. Kammel and S. Thrun, „Optimal trajectory generation for dynamic street scenarios in a Frenét Frame,“ in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 987–993, DOI: 10.1109/ROBOT.2010.5509799.

[319] A. Wischnewski, T. Stahl, J. Betz and B. Lohmann, „Vehicle dynamics state estimation and localization for high performance race cars,“ *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 154–161, 2019, DOI: https://doi.org/10.1016/j.ifacol.2019.08.064.

[320] J. Drosten, „Entwicklung und Implementierung einer visuellen Odometrie für autonome Rennfahrzeuge,“ Master's Thesis, Technische Universität München, München, 2021.

[321] L. Baierlein, „Entwicklung eines Lokalisierungsalgorithmus unter Verwendung bekannter Streckeninformationen,“ Master's Thesis, Technische Universität München, München, 2021.

[322] F. Brunhuber, „Synthetische Sensordaten für das Autonome Fahren: Entwicklung einer Simulationsumgebung für Autonome Rennfahrzeuge,“ Master's Thesis, Technische Universität München, München, 2021.

[323] S. Zhang, L. Zheng and W. Tao, „Survey and Evaluation of RGB-D SLAM,“ *IEEE Access*, vol. 9, pp. 21367–21387, 2021, DOI: 10.1109/ACCESS.2021.3053188.

[324] M. Rudolph, „Implementation of Depth Maps Generated from LiDAR Data in a Visual-SLAM,“ Master's Thesis, Technische Universität München, München, 2021.

[325] M. Hu, S. Wang, B. Li, S. Ning, L. Fan and X. Gong, „PENet: Towards Precise and Efficient Image Guided Depth Completion,“ in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13656–13662, DOI: 10.1109/ICRA48506.2021.9561035.

[326] Y. Long, D. Morris, X. Liu, M. Castro, P. Chakravarty and P. Narayanan, „Radar-Camera Pixel Depth Association for Depth Completion,“ in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 12502–12511, DOI: 10.1109/CVPR46437.2021.01232.

[327] J. Lambert, Z. Liu, O. Sener, J. Hays and V. Koltun, „MSeg: A Composite Dataset for Multi-Domain Semantic Segmentation,“ in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2876–2885, DOI: 10.1109/CVPR42600.2020.00295.

[328] O. Ronneberger, P. Fischer and T. Brox, „U-Net: Convolutional Networks for Biomedical Image Segmentation,“ in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015, pp. 234–241, ISBN: 978-3-319-24574-4.

[329] J. Yang, L. An, A. Dixit, J. Koo and S. I. Park, „Depth Estimation with Simplified Transformer,“ *arXiv preprint arXiv:2204.13791*, 2022.

[330] D. Hendrycks and K. Gimpel, „Gaussian error linear units (gelus),“ *arXiv preprint arXiv:1606.08415*, 2016.

[331] L. N. Smith and N. Topin, „Super-convergence: Very fast training of neural networks using large learning rates,“ in *Artificial intelligence and machine learning for multi-domain operations applications*, 2019, pp. 369–386, DOI: https://doi.org/10.1117/12.2520589.

[332] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, „Focal Loss for Dense Object Detection,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020, DOI: 10.1109/TPAMI.2018.2858826.

[333] L. Connert, „SEntwicklung eines Multi-Task-Learning-Modells zur Tiefenkartenerstellung mit monokularen Bildern und spärlichen Radardaten,“ Master's Thesis, Technische Universität München, München, 2022.

[334] C.-C. Lo and P. Vandewalle, „RCDPT: Radar-Camera Fusion Dense Prediction Transformer,“ in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5, DOI: 10.1109/ICASSP49357.2023.10096129.

[335] J.-T. Lin, D. Dai and L. V. Gool, „Depth Estimation from Monocular Images and Sparse Radar Data,“ in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10233–10240, DOI: 10.1109/IROS45743.2020.9340998.

[336] W.-Y. Lee, L. Jovanov and W. Philips, „Semantic-guided radar-vision fusion for depth estimation and object detection,“ in *BMVC, the 32nd British Machine Vision Conference, Proceedings*, 2021, p. 13. Available: https://www.bmvc2021-virtualconference.com/.

[337] A. D. Singh, Y. Ba, A. Sarker, H. Zhang, A. Kadambi, S. Soatto, M. Srivastava and A. Wong, „Depth Estimation from Camera Image and mmWave Radar Point Cloud,“ in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 9275–9285, DOI: 10.1109/CVPR52729.2023.00895.

[338] M. Muja and D. G. Lowe, „Fast approximate nearest neighbors with automatic algorithm configuration.“ *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[339] B. D. Lucas and T. Kanade, „An iterative image registration technique with an application to stereo vision,“ in *IJCAI'81: 7th international joint conference on Artificial intelligence*, 1981, pp. 674–679.

[340] B. Ma, „Conception and Implementation of a Visual-Lidar-based SLAM Algorithm,“ Master's Thesis, Technische Universität München, München, 2021.

[341] T. A. Foundation. „Autoware,“ 2024. [Online]. Available: https://www.https://autoware.org/ [visited on 05/25/2024].

[342] M. Pielmeier, „Multi-LiDAR Pipeline for Robust Localization and Mapping of Autonomous Vehicles," Master's Thesis, Technische Universität München, München, 2023.

[343] C. Weiß, „HD-Mapping for Autonomous Driving: Leveraging Various Data Sources for Highly Precise Maps," Master's Thesis, Technische Universität München, München, 2023.

[344] C. Scheibl, „Robust LiDAR-based Localization and Mapping Pipeline lor Autonomous Vehicles: Implementation, Application, Validation," Master's Thesis, Technische Universität München, München, 2023.

[345] Forschungsgesellschaft für Straßen- und Verkehrswesen e.V. „*Richtlinien für die Anlage von Straßen*," 1996.

[346] I. Tier IV. „Vector Map Builder - Autoware Tools," 2024. [Online]. Available: https://tools.tier4.jp/feature/vector_map_builder_ll2/ [visited on 05/25/2024].

[347] S. Umeyama, „Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991, DOI: 10.1109/34.88573.

[348] Marvin S. White, Jr. and Patricia Griffin, „Piecewise Linear Rubber-Sheet Map Transformation," *The American Cartographer*, vol. 12, no. 2, pp. 123–131, 1985, DOI: 10.1559/152304085783915135.

[349] M. Leitenstern, „HD-Mapping for Autonomous Driving: Multi-Information Fusion for a Scalable Mapping Pipeline," Master's Thesis, Technische Universität München, München, 2023.

[350] Bayrische Vermessungsverwaltung. „*OPENDATA: Kostenfreie Geodaten der Bayrischen Vermessungsverwaltung*," 2023. Available: https://geodaten.bayern.de/opengeodata/ [visited on 10/18/2023].

[351] H. Wang, C. Xue, Y. Tang, W. Li, F. Wen and H. Zhang, „LTSR: Long-term Semantic Relocalization based on HD Map for Autonomous Vehicles," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2171–2178, DOI: 10.1109/ICRA46639.2022.9811855.

[352] J. Wang, L. Zhang, Y. Huang, J. Zhao and F. Bella, „Safety of autonomous vehicles," *Journal of advanced transportation*, vol. 2020, pp. 1–13, 2020, DOI: https://doi.org/10.1155/2020/8867757.

[353] C. Chen, B. Wang, C. X. Lu, N. Trigoni and A. Markham, „Deep Learning for Visual Localization and Mapping: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2023, DOI: 10.1109/TNNLS.2023.3309809.

[354] M. Jung, W. Yang, D. Lee, H. Gil, G. Kim and A. Kim, „HeLiPR: Heterogeneous LiDAR Dataset for inter-LiDAR Place Recognition under Spatial and Temporal Variations," *arXiv preprint arXiv:2309.14590*, 2023.

Prior Publications

During the development of this dissertation, publications and student theses were written in which partial aspects of this work were presented.

Journals; Scopus/Web of Science listed (peer-reviewed)

[11] **F. Sauerbeck**, L. Baierlein, J. Betz and M. Lienkamp, „A Combined LiDAR-Camera Localization for Autonomous Race Cars,“ *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 12-05-01-0006, pp. 61–71, 2022, DOI: https://doi.org/10.4271/12-05-01-0006.

[12] **F. Sauerbeck**, S. Huch, F. Fent, P. Karle, D. Kulmer and J. Betz, „Learn to See Fast: Lessons Learned from Autonomous Racing on How to Develop Perception Systems,“ *IEEE Access*, pp. 1–1, 2023, DOI: https://doi.org/10.1109/ACCESS.2023.3272750.

[13] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp, B. Lohmann, F. Nobis, L. Ögretmen, M. Rowold, **F. Sauerbeck**, T. Stahl, R. Trauth, F. Werner and A. Wischnewski, „TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge,“ *Journal of Field Robotics*, 2022, DOI: https://doi.org/10.1002/rob.22153.

[16] **F. Sauerbeck**, D. Halperin, L. Connert and J. Betz, „CamRaDepth: Semantic Guided Depth Estimation Using Monocular Camera and Sparse Radar for Automotive Perception,“ *IEEE Sensors Journal*, 2023, DOI: 10.1109/JSEN.2023.3321886.

Conferences, Periodicals; Scopus/Web of Science listed (peer-reviewed)

[14] **F. Sauerbeck**, B. Obermeier, M. Rudolph and J. Betz, „RGB-L: Enhancing Indirect Visual SLAM Using LiDAR-Based Dense Depth Maps,“ in *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*, 2023, pp. 95–100, DOI: https://doi.org/10.1109/ICCCR56747.2023.10194045.

[18] **F. Sauerbeck**, D. Kulmer, M. Leitenstern, C. Weiss and J. Betz, „Multi-LiDAR Localization and Mapping Pipeline for Urban Autonomous Driving,“ (in press), in *2023 IEEE Sensors*, 2023.

[20] S. Huch, **F. Sauerbeck** and J. Betz, „DeepSTEP - Deep Learning-Based Spatio-Temporal End-To-End Perception for Autonomous Vehicles,“ in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–8, DOI: https://doi.org/10.1109/IV55152.2023.10186768.

[55] A. Kulkarni, J. Chrosniak, E. Ducote, **F. Sauerbeck**, A. Saba, U. Chirimar, J. Link, M. Cellina and M. Behl, „RACECAR–The Dataset for High-Speed Autonomous Racing,“ (in press), in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

Journals, Conferences, Periodicals, Reports, Conference Proceedings and Poster, etc.; not Scopus/Web of Science listed

[307] A. Wischnewski, M. Geisslinger, J. Betz, T. Betz, F. Fent, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, F. Nobis, L. Ögretmen, M. Rowold, **F. Sauerbeck**, T. Stahl, R. Trauth, M. Lienkamp and B. Lohmann, „Indy autonomous challenge-autonomous race cars at the handling limits,“ in *12th International Munich Chassis Symposium 2021: chassis. tech plus*, 2022, pp. 163–182.

J. Betz, H. Zheng, Z. Zang, **F. Sauerbeck**, K. Walas, V. Dimitrov, M. Behl, R. Zheng, J. Biswas, V. Krovi and R. Mangharam, „Teaching autonomous systems hands-on: Leveraging modular small-scale hardware in the robotics classroom,“ (in review process), *arXiv preprint arXiv:2209.11181*, 2022.

Non-thesis-relevant publications; Scopus/Web of Science listed (peer-reviewed)

T. Herrmann, **F. Sauerbeck**, M. Bayerlein, J. Betz and M. Lienkamp, „Optimization-based real-time-capable energy strategy for autonomous electric race cars,“ *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 12-05-01-0005, pp. 45–59, 2022, DOI: https://doi.org/10.4271/12-05-01-0005.

T. Dwivedi, T. Betz, **F. Sauerbeck**, P. Manivannan and M. Lienkamp, „Continuous Control of Autonomous Vehicles using Plan-assisted Deep Reinforcement Learning,“ in *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*, 2022, pp. 244–250, DOI: https://10.23919/ICCAS55662.2022.10003698.

P. Karle, F. Fent, S. Huch, **F. Sauerbeck** and M. Lienkamp, „Multi-Modal Sensor Fusion and Object Tracking for Autonomous Racing,“ *IEEE Transactions on Intelligent Vehicles*, pp. 1–13, 2023, DOI: https://10.1109/TIV.2023.3271624.

Thesis-relevant open-source software

[15] **F. Sauerbeck** and M. Rudolph. „GitHub | ORB_SLAM3_RGBL,“ 2023. [Online]. Available: https://github.com/TUMFTM/ORB_SLAM3_RGBL [visited on 07/08/2023].

[17] **F. Sauerbeck** and D. Halperin. „GitHub | CamRaDepth,“ 2023. [Online]. Available: https://github.com/TUMFTM/CamRaDepth [visited on 07/08/2023].

[19] M. Leitenstern and **F. Sauerbeck**. „GitHub | Lanelet2_OSM_Fusion,“ 2023. [Online]. Available: https://github.com/TUMFTM/Lanelet2_OSM_Fusion [visited on 10/24/2023].

Supervised Student Theses

The following student theses were written within the framework of the dissertation under the supervision of the author in terms of content, technical and scientific support as well as under relevant guidance of the author. In the following, the bachelor, semester and master theses relevant and related to this dissertation are listed. Many thanks to the authors of these theses for their extensive support within the framework of this research project.

[315] D. Kulmer, „Combined Map Based Localization and Object Detection for Autonomous Race Cars,“ Master's Thesis, Technische Universität München, München, 2022.

[320] J. Drosten, „Entwicklung und Implementierung einer visuellen Odometrie für autonome Rennfahrzeuge,“ Master's Thesis, Technische Universität München, München, 2021.

[321] L. Baierlein, „Entwicklung eines Lokalisierungsalgorithmus unter Verwendung bekannter Streckeninformationen,“ Master's Thesis, Technische Universität München, München, 2021.

[322] F. Brunhuber, „Synthetische Sensordaten für das Autonome Fahren: Entwicklung einer Simulationsumgebung für Autonome Rennfahrzeuge,“ Master's Thesis, Technische Universität München, München, 2021.

[324] M. Rudolph, „Implementation of Depth Maps Generated from LiDAR Data in a Visual-SLAM,“ Master's Thesis, Technische Universität München, München, 2021.

[333] L. Connert, „SEntwicklung eines Multi-Task-Learning-Modells zur Tiefenkartenerstellung mit monokularen Bildern und spärlichen Radardaten,“ Master's Thesis, Technische Universität München, München, 2022.

[340] B. Ma, „Conception and Implementation of a Visual-Lidar-based SLAM Algorithm,“ Master's Thesis, Technische Universität München, München, 2021.

[342] M. Pielmeier, „Multi-LiDAR Pipeline for Robust Localization and Mapping of Autonomous Vehicles,“ Master's Thesis, Technische Universität München, München, 2023.

[343] C. Weiß, „HD-Mapping for Autonomous Driving: Leveraging Various Data Sources for Highly Precise Maps,“ Master's Thesis, Technische Universität München, München, 2023.

[344] C. Scheibl, „Robust LiDAR-based Localization and Mapping Pipeline lor Autonomous Vehicles: Implementation, Application, Validation,“ Master's Thesis, Technische Universität München, München, 2023.

[349] M. Leitenstern, „HD-Mapping for Autonomous Driving: Multi-Information Fusion for a Scalable Mapping Pipeline,“ Master's Thesis, Technische Universität München, München, 2023.

E. Herrmann, „Implementation and Comparison of Different Localization Algorithms for Autonomous RC Vehicles,“ Bachelor's Thesis, Technische Universität München, München, 2020.

B. Obermeier, „Development of an Automated Evaluation Pipeline for Depth Enhanced Visual SLAM,“ Bachelor's Thesis, Technische Universität München, München, 2022.

M. Ribeiro de Oliveira, „Full-Stack Software Analysis of Autonomous Small-Scale Race Cars,“ Bachelor’s Thesis, Technische Universität München, München, 2022.

F. Toth, „Development of a Map Builder and Localization for the Autonomous Vehicle muc022,“ Bachelor’s Thesis, Technische Universität München, München, 2022.

N. Morbitzer, „Self-supervised monocular depth estimation in all conditions,“ Interdisciplinary Project, Technische Universität München, München, 2023.

S. San Antonio de Benito, „Implementierung und Validierung eines multimodalen SLAM Algorithmus,“ Master’s Thesis, Technische Universität München, München, 2022.

X. Luo, „Development of a Front-End Module for a Decentralized Multi-Modal SLAM Framework in the Domain of Mobile Robotic Applications,“ Master’s Thesis, Technische Universität München, München, 2022.

K. Farkas, „Development and Implementation of a HD-Mapping Toolchain for Urban Regions,“ Master’s Thesis, Technische Universität München, München, 2023.

F. Jahncke, „Sensor Fusion-Based Perception for an Autonomous Racecar,“ Master’s Thesis, Technische Universität München, München, 2023.

E. Keras, „Vehicle Dynamics State Estimation and Sensor Fusion for Urban Autonomous Driving,“ Master’s Thesis, Technische Universität München, München, 2023.

M. Spitzar, „Echtzeitfähige Deep Learning-basierte Generierung von Tiefenkarten,“ Master’s Thesis, Technische Universität München, München, 2022.

D. Halperin, „Semantic Guided Depth Estimation Using Monocular Camera,“ Master’s Thesis, Technische Universität München, München, 2023.

M. Domsch, „Sensor and Environment Simulation of an Autonomous Research Vehicle,“ Semesterarbeit, Technische Universität München, München, 2023.

K. Eckert, „Evaluation kartenbasierter Lokalisierungsmethoden für autonome Rennfahrzeuge,“ Semesterarbeit, Technische Universität München, München, 2021.

B. Ma, „Development of an Algorithm to Calculate the Referenceline on Different Racetracks,“ Semesterarbeit, Technische Universität München, München, 2020.

M. Spitzar, „Entwicklung eines Tools zur Kalibrierung verschiedener Sensoren eines autonomen Rennwagens,“ Semesterarbeit, Technische Universität München, München, 2021.

C. Weiß, „Implementierung eines LiDAR-SLAM für die Indy Autonomous Challenge,“ Semesterarbeit, Technische Universität München, München, 2021.