UNIVERSITÄT BONN

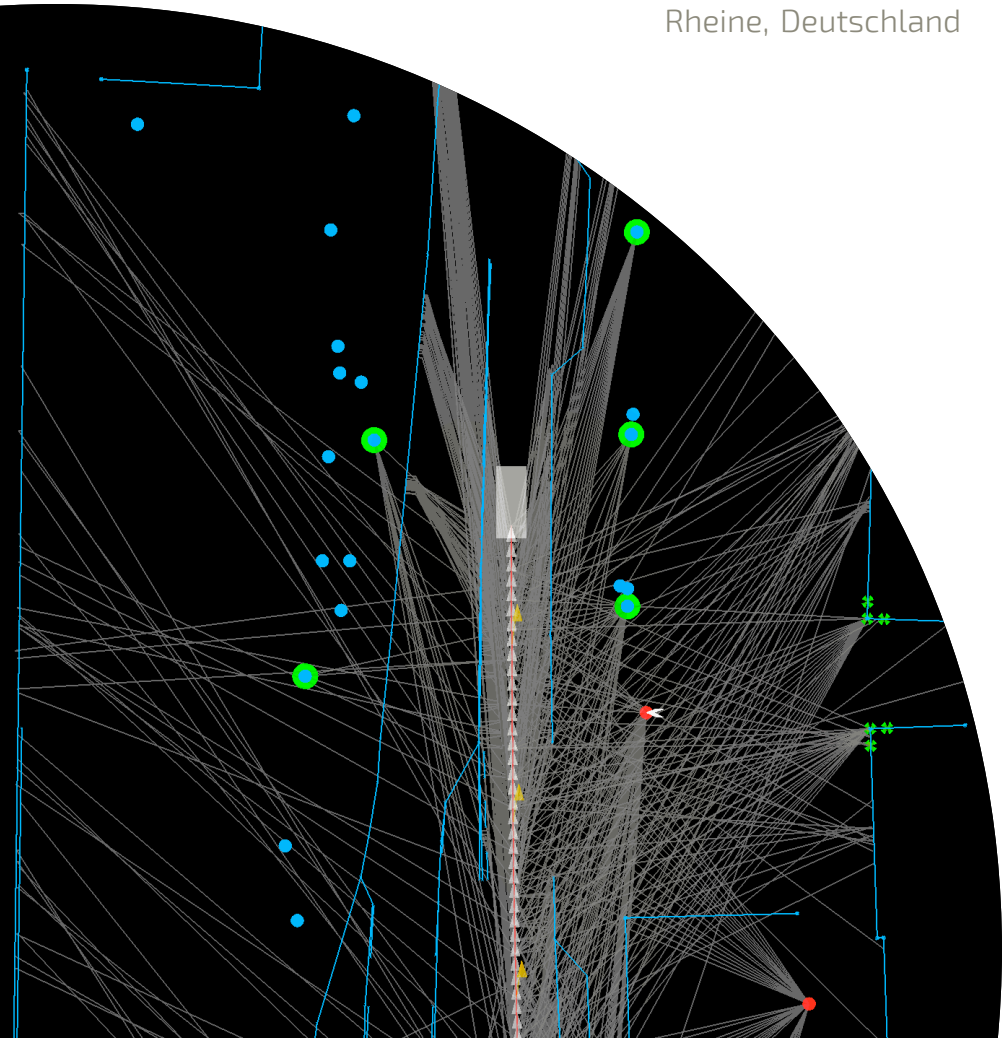
Inaugural-Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Landwirtschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn
Institut für Geodäsie und Geoinformation

Graph-based Sliding Window Localization and Map Refinement for Automated Vehicles

von

Daniel Wilbers

aus
Rheine, Deutschland



Bonn 2021

Referent:
Prof. Dr. Cyrill Stachniss, University of Bonn, Germany

Korreferent:
Prof. Dr. Ingmar Posner, University of Oxford, UK

Tag der mündlichen Prüfung: 24. September 2021

Angefertigt mit Genehmigung der Landwirtschaftlichen Fakultät der Universität Bonn

Zusammenfassung

Lokalisierung ist ein wesentlicher Bestandteil von automatisierten und vollautonomen Fahrzeugen. Die genaue Position eines Fahrzeuges wird benötigt, um Informationen über die Umgebung aus Karten auslesen zu können. Diese Zusatzinformationen, wie z. B. Verkehrsregeln, werden während der Fahrt als Ergänzung zur Sensordateninterpretation genutzt, um Funktionen, wie Szenenverständnis und Trajektorienplanung, zu unterstützen. Thema dieser Arbeit ist die Entwicklung eines Lokalisierungssystems für das automatisierte Fahren in urbanen Umgebungen. Der vorgestellte Ansatz nutzt markante Umgebungsobjekte, sogenannte Landmarken, die vom Fahrzeug detektiert und mit einer Karte abgeglichen werden, um das Auto zu lokalisieren. Eine besondere Herausforderung ist hierbei die Nutzung von Drittherstellerkartenmaterial. Diese sind üblicherweise nicht auf eine spezifische Sensorkonfiguration zugeschnitten. Dadurch besteht eine Abweichung zwischen den vom Fahrzeug detektierbaren und den in den Karten verzeichneten Landmarken, welche in dieser Arbeit explizit berücksichtigt wird. Dies ermöglicht den Einsatz von Drittherstellerkarten für Landmarkendetektionen auf Basis unterschiedlicher Sensorik. Eine weitere Herausforderung bei dem Einsatz von Karten zur Lokalisierung ist, dass diese im Laufe der Zeit veralten. Dies kann im Rahmen des automatisierten Fahrens sicherheitskritisch sein. Für einen langfristig sicheren Betrieb ist es daher notwendig, Karten regelmäßig zu aktualisieren. Der vorgestellte Ansatz trägt zu diesem Ziel bei, indem zur Laufzeit Update-Hypothesen berechnet und an ein Back-End zur Validierung mit Hilfe von Flottendaten übermittelt werden. Im Mittelpunkt steht hier ein Graphen-basiertes Sliding Window Optimierungsverfahren, welches sowohl die Trajektorie des Fahrzeuges als auch die Update-Hypothesen mit hinreichender Genauigkeit und Effizienz berechnet. Innerhalb der Arbeit wird erläutert, wie sich die Faktoren des Graphens bestimmen und die Karte als A-Priori-Verteilung einbinden lassen. Das Verfahren fusioniert Landmarkenmessungen unterschiedlicher Sensorik, wie z. B. LiDAR und Kamera, zusammen mit GNSS- und Odometriedaten. Zusätzlich wird ein neues Verfahren vorgestellt, welches erlaubt die Informationen von zeitlich veralteten Messungen im Optimierungsverfahren zu erhalten. Die vorgestellten Verfahren werden ausführlich anhand von realen Daten eines automatisierten Fahrzeuges evaluiert.

Abstract

Localization is an essential task in automated driving and advanced driver assistance systems. The precise location of a vehicle is required for extracting information from a map about the vehicle's environment. The map information is typically used to enrich the perceptions of the vehicle using its sensors to ease tasks such as scene understanding or planning. This thesis investigates the challenge of creating a localization framework for automated driving in urban scenarios. Our localization approach relies on detecting distinct elements, so-called landmarks, in the vehicle's surroundings and aligning them to a given map to infer the vehicle's location. A particular challenge that we investigate is using general-purpose landmark maps from third-party distributors for localization. These maps are typically not tailored towards a specific sensor setup or the vehicle's direct sensor configuration, such that the third-party map is not tuned to what the vehicle is able to detect. We explicitly consider this challenge within our approach. In return, our approach allows for using landmark detections from different sensors with a single general-purpose map for localization. Another fundamental challenge of map-based approaches is that maps are getting outdated over time, which is a safety-critical aspect of localization for automated driving. Therefore, it is necessary to establish a reliable update process for successful long-term operation. We contribute to that goal by computing map update hypotheses during operation that could be transmitted to a server back-end for fleet-based validation. We solve localization and map refinement in a combined approach that uses graph-based sliding window optimization to estimate the vehicle's trajectory and landmark positions in an accurate and computationally efficient way. We describe how to construct our approach as a factor graph and derive its necessary factors to model the map landmarks as a prior over the landmark detections. Our approach fuses landmark detections from various sensors, e.g., LiDAR and camera, together with odometry and GNSS measurements. Additionally, we present how to compute sparse global priors, a novel sparsification scheme that allows us to preserve information that we remove from our sliding window without the drawbacks of dense marginalization. We thoroughly evaluate our approach based on real-world data and trajectories recorded in an automated vehicle.

Acknowledgments

First of all, I would like to express my gratitude for receiving the chance to write this thesis as an external Ph.D. student of the University of Bonn at Volkswagen Group Innovation. For that, I would like to thank my supervisor Professor Cyrill Stachniss for his excellent support and guidance. I am especially grateful for giving me the freedom that allowed me to flourish my own ideas, as well as the fruitful and open discussions. I could not have hoped for better supervision.

I would like to thank Christian Merfels, without whom I would not have had the opportunity to write this thesis in the first place. Our discussions were always a pleasure and vastly helped me in producing new concepts and realizing them. Thank you very much to the whole Photogrammetry & Robotics Lab. I always felt welcome in Bonn and enjoy remembering meeting everyone at conferences. Special thanks to Olga Vysotska for proof-reading my papers and parts of my thesis. I deeply appreciated the open and honest feedback. My sincere thanks to Birgit Klein for her kind and caring support.

Likewise, I would like to thank my colleagues and friends Bernd Rech, Niklas Koch, Constanze Hungar, Christian Merfels, Thilo Schaper, Stefan Jürgens, Christopher Plachetka, Jenny Fricke, and Andreas Kwoczek not only for their support but also for making my Ph.D. time a true pleasure. Special thanks to Bernd Rech for providing his professional perspective and valuable advice.

Additional thanks to my colleagues and supervisors at Volkswagen Group Research for keeping up the team spirit and providing a friendly work environment. Many thanks to Henrik Bohlke and Lars Rumberg for collaborating with me and the group.

Most heartwarmingly, I would like to thank Juliane Spieker for always being there for me.

Last but not least, I would like to deeply thank my family for their unconditional care and support since I can remember.

Disclaimer

The results, opinions, or conclusions of this dissertation are not necessarily those of the Volkswagen AG.

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

Contents

Chapter 1

Introduction

Automated driving is a technology that will have a drastic impact on our everyday life. While the applications span from individual and public passenger transportation to industrial logistics, the benefits of automated vehicles are numerous. On a macroscale, the main public advantages are increased road safety for traffic participants and potentially more environment-friendly driving. Besides that, on a microscale, individual benefits include increased mobility, efficiency, and comfort.

Starting decades ago with simply the vision in mind, researchers have tackled many different aspects of automated driving (e.g., Tsugawa et al. (1979) and Dickmanns et al. (1990)). Nowadays, advanced driver assistance systems (ADAS) with a limited level of autonomy, like automated emergency braking and assisted steering, are already commercially available (e.g., Volkswagen AG (2017) and Tesla, Inc. (2018)). Nevertheless, with the increasing level of autonomy, new challenges arise that still must be solved. As the responsibility of the system grows, automated driving functions must be more failsafe, more robust, and more reliable than before. This imposes new requirements on hard- and software, which are being tackled in recent research.

A promising approach for alleviating the challenges of software modules involved in automated driving is to use knowledge based on a given map. If the vehicle knows its location relative to a map, it can enrich its environment model with information from this map. For example, in blocked line-of-sight scenarios, objects like crosswalks and traffic lights that can not be detected directly with the vehicle's sensors can still be incorporated into the decision-making by extracting them from a map. Comparable to the experience of a human driver, map-based approaches model prior knowledge about the environment and thus help to overcome sensor limitations. Hence, improved anticipatory driving is possible, which contributes to the safety and reliability of automated driving. For map-based approaches, it is crucial that the vehicle can estimate its location within the map.

This ability is referred to as self-localization and is the main focus of this thesis.

1.1 Localization for automated driving

Besides the conventional use of self-localization in navigation systems, knowing the vehicle's location is a prerequisite for many different applications. For example, it is used in augmented reality windscreens that provide information about the vehicle surrounding to the passenger, as well as in cooperative driving functions. Within automated driving, the vehicle's location is frequently used for perception, prediction, and trajectory planning, which all benefit from using information from a given map.

In general, a vehicle's location is defined as its pose, which is the vehicle's position and orientation. In the context of this thesis, we perform map-relative localization on a globally accurate map, such that our estimated pose is also in a global reference frame. Compared to other mobile robotic applications, where the coordinate system is usually local, using a global frame is preferable in automated driving. It allows the direct substitution or supplementation of other global localization approaches (e.g., satellite-based localization). Besides, it eases the integration of other globally referenced information sources, like other complementary navigation maps or traffic data.

A popular approach to global localization is using one or multiple of the available Global Navigation Satellite Systems (GNSS). Depending on the region, different variants are applicable. Besides GLONASS, BeiDou, Galileo, and QZSS, the most common one is the Global Positioning System (GPS). A classical GPS setup with one antenna is able to provide a position estimate within a meter-level accuracy. Using a real-time kinematic (RTK) system with two antennas drastically improves the accuracy to a centimeter-level and additionally provides the vehicle's orientation. This is achieved by incorporating atmospheric correction data provided by a nearby base station. Nevertheless, a major drawback of satellite-based approaches is that they only work optimally in open sky scenarios. Whenever multipath or blocked line-of-sight effects occur, the achieved accuracy and precision are typically heavily degraded. In extreme cases, localization might even not be possible at all. As this is typical in urban scenarios, like parking garages and street canyons, localization with GNSS alone is not sufficient for automated driving.

Another well-known approach to localization is called landmark-based localization, which has a long tradition, e.g., through geodetic observing towers built for triangulation networks (Agarwal et al., 2014a). Thrun et al. define a landmark as '[…] distinct, stationary features of the environment that can be recognized reliably' (Thrun et al. 2005, p. 21). The general idea is to detect landmarks at

runtime in the vehicle and match the set of observed landmarks to a predefined landmark map. Whenever the map is globally referenced, so is the inferred vehicle pose. In that case, landmark-based localization can directly be used complementary or supplementary to GNSS. To detect landmarks in the first place, different kinds of sensor technologies are applicable. The most common ones are LiDAR, radar, and camera. Incorporating multiple sensing modalities in a single localization approach is crucial for the robustness against sensor faults. It enables the system to robustly work in different environment scenarios, like highways and cities.

A requirement within automated driving is that localization systems must work at first operation. For landmark-based localization this implies that the globally referenced landmark map must be available at startup and can not be created on-the-fly. Nowadays, an initial map is usually created in a semi-automated process that requires a special mapping drive with another vehicle. Creating landmark maps still involves much manual effort to filter out dynamic objects and label data, which is why it is resource expensive. Comparable to navigation maps, it is necessary that only one common initial map is required for a whole fleet of vehicles. Ideally, the initial map can be used in a general-purpose fashion in combination with various sensors, which we investigate in this thesis. Perspectively, third-party mapping companies will be able to provide these highly accurate initial landmark maps on a large scale. These third-party landmark maps are typically not tailored towards a specific sensor setup or a vehicle's sensor configuration but are instead created as general-purpose maps. As an effect, the elements of general-purpose third-party maps are not tuned to what a specific vehicle is able to detect with its sensors, which is a challenge that we investigate within this thesis.

A fundamental challenge of any map-based localization is that they suffer from maps getting outdated over time. Leading to degraded accuracy and possibly complete failure, maintaining the landmark map is a key challenge for the long-term functionality of landmark-based localization systems. One approach for updating maps is to rely on over-the-air updates, which are either based on fleet data or manually engineered. To keep updating maps manageable in large-scale applications, it is beneficial to compute the map updates in an automated fashion. A requirement for this is that vehicles can transfer their recorded data to a back-end service. Thereby the amount of data that needs to be transferred is ideally as small as possible. The most expressive form would be to transmit the raw measurements to the back-end, which is unfeasible for over-the-air communication due to the large amount of data. As a part of this thesis, we investigate the challenge of estimating hypothesis for landmark updates directly on-board within a vehicle. Our approach reduces the amount of data transmitted to a back-end.

In general, a localization system for automated driving must be designed to

- work in a variety of scenarios independent of the environment,
- applicable to a broad range of sensor setups,
- support third-party maps,
- and provide long-term operability through refining its map,

which brings us to the key question of this thesis: How can we design a multimodal landmark-based localization system that works with third-party maps and can estimate in-vehicle map refinements?

1.2 Scope and limitations

In this thesis, we investigate how to design a landmark-based localization system for automated driving. We utilize graph-based optimization techniques for estimating the vehicle trajectory within a sliding window and explain the necessary steps for the state estimation. This covers an in-depth description of how we solve data association, construct the graph, and define the involved error functions. Our center of interest is on adapting graph-based optimization to a sliding window paradigm such that it is computationally fast enough for automated driving and incorporates general-purpose third-party landmark maps. In addition, we investigate how to incorporate measurement data outside of the sliding window through marginalization.

We do not investigate the optimization itself but instead rely on well-known algorithms. We discuss the advantages and disadvantages of graph-based optimization compared to particle filters and evaluate both approaches against each other.

Our system does not rely on specific sensors for landmark measurements but instead incorporates landmarks in a general way, which allows us to incorporate measurements from various sensor sources flexibly. Mainly these include landmarks from LiDAR, radar, and camera. We do not investigate how to design a detector for landmarks but rely on already available modules. This specifically means that we cope with landmark detectors as if they are black boxes, which is beneficial for easily changing detector modules with different sensor setups and vehicles. Although the concepts presented in this thesis are in general applicable to 3D poses, we limit ourselves to estimating the vehicle's global 2D pose. This is sufficient for our use case of automated driving in urban scenarios as our vehicle moves on a ground plane and map information is stored relative to the ground.

While our primary focus is on the localization itself, i.e., estimating the global vehicle pose, we also investigate how to estimate the position of landmarks for

delayed map refinements. We use the latter as update hypotheses that can be transmitted to a remote back-end service, which in this context is an application running in the cloud. We conceptually discuss how a back-end side could be integrated into our localization system but do not cover the remote back-end service itself. In the context of automated driving, it is essential to minimize the risk of defective updates, which could potentially have a safety-critical impact. Therefore, we only update the map after a back-end service has validated the update hypotheses. This is in contrast to standard simultaneous localization and mapping (SLAM) systems that continuously generate and update a map. We use the term *delayed map refinement* to emphasize that we do not immediately update the map but first conceptually wait for back-end validation.

In detail, our localization system must be able to fulfill a set of requirements to be applicable for automated driving. Our localization system must be

- highly accurate, whereas the intended minimum average Euclidean error depends on the driving scenario,
- incorporate landmarks in a universal way, such that multiple sensor modalities can easily be integrated,
- computationally tractable for online localization,
- provide pose estimates with a frequency between 10 Hz to 20 Hz, depending on the scenario,
- rely on general-purpose landmark maps whenever available,
- fallback to a general-purpose fusion of GNSS and odometry otherwise,
- and estimate conservative in-vehicle map refinements as update hypotheses that can be transmitted to a back-end service.

In the remainder of this thesis, we investigate how to fulfill these requirements.

1.3 Main contributions

The main contribution of this thesis is the design of a graph-based localization system suitable for automated driving addressing the criteria mentioned above.

At first, we contribute to adapting graph-based optimization such that it is suitable for automated driving. We show how to separate data association into local association between landmark measurements and global association to a given general-purpose map. Together with a temporal association smoothing, our system can successfully revise map associations and perform delayed association. In relation to computational tractability, we employ a sliding window over

vehicle poses that limits the size of the state vector. In return, this limits the computation time required for our localization approach. Overall, our architecture contributes to having a frequent and globally accurate self-localization. We analyze our approach's properties and provide an in-depth evaluation with the main focus on urban scenarios.

Second, we contribute to using general-purpose third-party landmark maps within localization for automated driving. We introduce a map matching algorithm that allows us to use landmark maps that are not tailored to a specific sensor setup. This is essential for the flexibility of our localization system. It allows us to incorporate landmark measurements from various sensors without adjusting our localization approach.

Third, we examine how to limit the loss of information when removing measurements from the sliding window. We derive sparse global priors that approximate dense marginalization to capture removed information. Our approach avoids marginalization fill-in in the graph and has the same sparsity pattern as without marginalization. Additionally, we show how to utilize global linearization points instead of local ones and evaluate the benefits.

Fourth, we show that our sparse global prior approach contributes to computing conservative estimates for landmark updates with our online system. Our focus is on computing update hypotheses, which are send to a back-end service.

Fifth, we derive the error functions required for different types of landmarks. This includes point landmarks and polyline-based landmarks, which require particular focus in relation to having a full rank optimizable system and ambiguities. Our contribution is the comparison between the different concepts of landmark integration.

1.4 Publications

Parts of this thesis have been published in the following peer-reviewed conference and patent applications:

- Daniel Wilbers, Christian Merfels, and Cyrill Stachniss. A Comparision of Particle Filter and Graph-based Optimization for Localization with Landmarks in Automated Vehicles. In *Proceedings of the IEEE International Conference on Robotic Computing (IRC)*, 2019b

- Daniel Wilbers, Lars Rumberg, and Cyrill Stachniss. Approximating Marginalization with Sparse Global Priors for Sliding Window SLAM-Graphs. In *Proceedings of the IEEE International Conference on Robotic Computing (IRC)*, 2019c

- Daniel Wilbers, Christian Merfels, and Cyrill Stachniss. Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019a

Patent applications:

- Positionsbestimmungssystems für eine mobile Einheit. *Patent application at Deutsches Patent- und Markenamt, Germany*, DE 10 2018 117 660.0
- Positionsbestimmungssystem und Verfahren zum Betreiben eines Positionsbestimmungssystems für eine mobile Einheit. *Patent application at Deutsches Patent- und Markenamt, Germany*, DE 10 2018 133 461.3
- Verfahren zur Aktualisierung einer Umgebungskarte, Vorrichtung für die fahrzeugseitige Durchführung von Verfahrensschritten des Verfahrens, Fahrzeug, Vorrichtung für die zentralrechnerseitige Durchführung von Verfahrensschritten des Verfahrens sowie computerlesbares Speichermedium. *Patent application at Deutsches Patent- und Markenamt, Germany*, DE 10 2018 118 215.5
- Verfahren zur Schätzung der Lokalisierungsgüte bei der Eigenlokalisierung eines Fahrzeuges, Vorrichtung für die Durchführung von Verfahrensschritten des Verfahrens, Fahrzeug sowie Computerprogramm. *Patent application at Deutsches Patent- und Markenamt, Germany*, DE 10 2018 118 220.1
- Verfahren zum Bewerten einer digitalen Karte, sowie Bewertungssystem. *Patent application at Deutsches Patent- und Markenamt, Germany*, DE 10 2020 115 743.6
- Verfahren zum Beurteilen einer Genauigkeit einer Positionsbestimmung einer Landmarke, sowie Bewertungssystem. *Patent application at Deutsches Patent- und Markenamt, Germany*, DE 10 2020 115 746.0

Chapter 2

Related work

In this chapter, we summarize the current state of the art and outline the key points of the main related work relevant to this thesis. We start by relating our approach to general vehicle localization in Section 2.1. Afterward, we discuss using GNSS for localization in relation to our work in Section 2.2. Furthermore, we review perception-based localization approaches in relation to their sensor and landmark types in Section 2.3. In Section 2.4, we focus on state estimation techniques for localization. In more detail, we provide an overview of robust estimation techniques in Section 2.4.1, sliding window approaches in Section 2.4.2, and marginalization and sparsification approaches related to our map refinement approach in Section 2.4.3. We categorize the different types of maps used for localization in Section 2.5 and discuss techniques for landmark map refinement in Section 2.5.3.

2.1 Vehicle localization

Estimating a vehicle's pose is required for many different applications in, e.g., aerial, maritime, underwater, indoor, ground, and other environments. While the imposed challenges on sensors are substantially different, similar mathematical backgrounds and concepts can be found across all domains. Among the most common localization techniques for vehicle localization are satellite-aided methods, ground-based infrastructure techniques (e.g., cell-based, Wi-Fi-based, ultra-wideband-based, and Bluetooth-based approaches), and map-based approaches that rely on environmental perception. We refer to the survey of Kuutti et al. (2018) for a review of the various technologies commonly used in localization for autonomous driving. In addition to our review, it contains vehicle-to-everything (V2X) localization approaches and provides further information about the different technologies' working principles. In contrast to GNSS and map-based localization, vehicle-to-vehicle approaches highly depend on the number of par-

ticipating surrounding vehicles and vehicle-to-infrastructure approaches on the availability of costly infrastructure (Kuutti et al., 2018). For these reasons, V2X approaches are unsuitable for scalable localization within the near future. Therefore, we focus on GNSS and perception-based localization approaches. Compared to GNSS-based approaches performing global localization, approaches based on environment perception estimate poses relative to a map. Nevertheless, map-relative localization approaches can also be used for global localization whenever the map is given in global coordinates. The prevalent idea in perception-based localization is to infer the vehicle location by matching perceived environment elements to a map. An advantage of perception-based approaches for automated driving applications is that they allow self-localization in areas with limited or no satellite visibility, e.g., in parking garages (Houben et al., 2015). Still, the ability of GNSS to directly compute global poses and its worldwide availability make it a valuable asset for automated driving. We review GNSS-based localization approaches and the challenges concerning automated driving in urban areas in the following section and afterward review perception-based localization techniques.

2.2 GNSS-based localization

The usage of GNSS-based localization has been investigated and enhanced for many years. Nowadays, GNSS variants like GLONASS, BeiDou, Galileo, QZSS, GPS are available and commonly used in, e.g., smartphones, navigation systems, and many more applications. We refer to Teunissen and Montenbruck (2017) for a comprehensive GNSS overview, including principles, variants, augmentation techniques, related algorithms, and applications. In the following, we focus on the related aspects of GNSS w.r.t. localization for automated driving in urban environments.

While classical GPS setups with a single antenna achieve meter-level accuracy and only provide estimates of the position, RTK techniques with two antennas nowadays allow for centimeter-level accuracy and provide orientation estimates (U.S. Government - National Coordination Office for Space-Based Positioning, Navigation, and Timing, 2018). The main idea of augmentation techniques like RTK is to incorporate atmospheric correction data from nearby base stations. The potential of using RTK for SLAM tasks is, for example, demonstrated by Klingbeil et al. (2014) and Schneider et al. (2016). They present a computationally efficient and accurate approach that combines vision, IMU, and RTK-based GPS and is applicable to light-weight unmanned aerial vehicles. A limitation of GNSS approaches is that they only work optimally under open sky conditions. The more restricted the satellite visibility, the more challenging it is to compute accurate and reliable pose estimates. The dominant error sources are non-line-of-

sight effects, in which the direct view between satellite and receiver is obstructed, and multipath effects, in which satellite signals are reflected by, e.g., surrounding buildings. Breßler et al. (2016) and Ollander et al. (2018) review these effects and point out their importance for urban scenarios. Stephenson (2016) points out that an additional error source for RTK-based systems is poor cellular coverage, limiting receiving correction data. Nevertheless, cellular coverage is usually more critical in rural areas than in urban scenarios.

Using GNSS in automotive applications has a long history, mainly for navigation systems and, more recently, ADAS applications (Skog and Händel, 2009). A recent study that compares the performance of various GNSS receivers in road scenarios is presented by Štern and Kos (2018). Among other things, they compare current low-cost receivers and more expensive ones in the urban area of Nantes, France. Their main conclusion is that pure GNSS-based receivers are unsuitable for sub-meter localization accuracy in urban environments. They emphasize that multi-sensor fusion is required for more accurate localization. Joubert et al. (2020) survey the current state-of-the-art GNSS-based approaches w.r.t. automated driving. They state that lane-level accuracy on highways is achievable with inertial navigation system (INS)-aided GNSS approaches but underline that GNSS outages in urban scenarios remain an issue. In line with their findings, Reid et al. (2019b) examine the performance of RTK-aided GNSS solutions on North American highways and find that lane-level localization is achievable under favorable GNSS conditions. Likewise, their results show that passing through major urban centers with poor satellite reception and multipath effects is still challenging with GNSS-based approaches. For more details on the applicability of correction data services for the automotive industry, we refer to Vana et al. (2019). Zimmermann et al. (2015) investigate the effect of various satellite obstruction scenarios (e.g., urban canyons) and suspect that multipath effects and blocked line-of-sight have a more significant impact on the accuracy than a degraded satellite geometry. In the context of automated driving, the Stanley vehicle winning the 2005 DARPA Grand Challenge (Thrun et al., 2006) successfully used GPS for navigating on roads in a desert. The vehicle could compensate for GPS outages and shifts by including an Inertial Measurement Unit (IMU), wheel encoders, and different fusion models depending on the GPS status. In the later 2007 DARPA Urban Challenge, the Junior vehicle (Montemerlo et al., 2008) successfully used GPS in combination with detecting road reflectivities and curbs to overcome GPS inaccuracies in an urban scenario. Nowadays, the majority of localization approaches for automated driving applications rely on fusing multiple sensor modalities. Their targets range from overcoming short-term GNSS outages, e.g., by fusing in-vehicle sensor data, to fully independent GNSS operation. For an accurate and robust localization, most of these methods rely on fusing

environmental perception data, which we review in Section 2.3. In this thesis, we use GNSS for globally initializing our approach and show how to incorporate GNSS pose estimates in our sensor fusion framework. Also, we demonstrate the challenging conditions for GNSS in urban environments and show that our approach only requires GNSS for initialization and works without it otherwise.

2.3 Perception-based localization

In the following, we review approaches related to road vehicle localization categorized by their main sensor type for environment perception. We start with camera-based approaches followed by radar-based and LiDAR-based approaches. We conclude with techniques that incorporate multiple perception sensors. During our review, we focus on the different types of landmarks used for localization.

Using cameras for localization and mapping tasks, which is also known as visual simultaneous localization and mapping (vSLAM), is a broad and active research field with many applications and, for example, surveyed by Taketomi et al. (2017) and Saputra et al. (2018). Closely related to vSLAM is the field of visual odometry (VO), which can be seen as SLAM without loop closures (Cadena et al., 2016). We refer to Scaramuzza and Fraundorfer (2011) and Fraundorfer and Scaramuzza (2012) for a survey on VO techniques. In relation to our approach VO approaches can be used as an odometry input to our system.

In general, SLAM approaches, and thus also vSLAM, relate to our approach based on the state estimation technique and applied landmark types. We review the latter in the context of road vehicle localization for automated driving in the following and refer for a broader overview to Cadena et al. (2016). The approach by Ziegler et al. (2014) integrates two separate camera-based features. Their approach relies on visual point features for urban localization and uses camera-based lane and curb detections in rural areas. Another camera-based approach is presented by Harr et al. (2018), who use two separate rear and front-facing cameras to detect lane markings for localization on highways. Bürki et al. (2019) present an approach that uses four cameras in a surround-view system for extracting visual features. Another visual landmark type is used by Ranganathan et al. (2013), who perform global localization based on corner features of road markings and visual odometry. Their approach is demonstrated on a globally referenced and self-generated map in a parking lot and operates on camera only. The approach by Suhr et al. (2017) uses classified camera-based road markings as well as lane detections from a mono camera fused with IMU and wheel speed data for urban localization. Jeevan et al. (2010) present a GNSS-independent localization approach that uses the ground markings of parking spots for localization. Based on two cameras for stereo vision, Poggenhans et al. (2018) present an approach

that uses road markings detected in a generated top-view image. Spangenberg et al. (2016) detect pole landmarks based on front-facing stereo cameras and show that visual pole landmarks are applicable for urban localization. Caselitz et al. (2016) propose a mono camera-based approach that matches image features to a given 3D point cloud map. In this thesis, we rely on road marking detections from a front-facing camera and a top-view camera system, which we integrate besides other landmarks within our evaluation.

A growing research direction that could impact the quality of visual odometry and vSLAM approaches in the context of automated driving is event-based vision. Compared to frame-based cameras, event-based cameras have a different working principle such that conventional techniques are no longer applicable and new methods are required. For example, Vidal et al. (2018) introduce an approach for drone-based SLAM based on event cameras that works under challenging conditions like low-light and highly dynamic lighting. Gallego et al. (2020) review the current state-of-the-art of event-based vision and its applications. We refer to them for further reference. Despite recent improvements, the applicability of event-based vision for map-based localization in road vehicles is still an open field.

Another actively researched field is visual place recognition, which is, e.g., reviewed by Lowry et al. (2016). Most place recognition techniques aim to recognize the place that is shown in an image instead of directly estimating the camera pose. Nonetheless, approaches like Naseer and Burgard (2017) and Valada et al. (2018) show that the boundaries to conventional SLAM are vanishing. While visual place recognition techniques are essential in the context of loop closure detection for SLAM applications, their applicability to our localization use case is, however, limited. The main reason is that most place recognition approaches require a special type of map, e.g., an image database, which is limited in the transferability to other sensors. In comparison, we target to use general-purpose maps that are usable with different sensors. In addition, the global accuracy of visual place recognition is still limited, such that pure visual place recognition is unsuitable in the context of automated driving. Nevertheless, the latest improvements in handling appearance changes could be an asset for localization in the context of automated driving. Nowadays, visual place recognition tasks are able to cope with severe appearance changes through, e.g., illumination, weather, seasons, different viewpoints, and dynamic objects in the scene. Approaches presented by, e.g., Milford and Wyeth (2012), Sünderhauf et al. (2015), Vysotska and Stachniss (2016b), Schönberger et al. (2018), Naseer et al. (2018), Vysotska and Stachniss (2019), and Vysotska (2019), have all shown the importance of handling appearance changes specifically for road vehicle localization. In relation to our approach, visual place recognition could serve as an alternative to GNSS, which would especially be helpful in GNSS-denied regions. Visual place recognition ap-

proaches that extract landmarks could, in theory, also be used as an input to our approach. Most methods do not focus on geometric semantic landmarks but instead use image features, making them difficult to apply in our general-purpose approach.

Another commonly used sensor type within road vehicles are radar sensors. One of the advantages for ADAS applications is that radar sensors even robustly work in challenging weather conditions (Langer, 1996). For example, Cen and Newman (2018) highlight the benefits of radar-based landmarks for ego-motion estimation under adverse conditions like rain and darkness. They also extend and evaluate their approach for urban and other scenarios (Cen and Newman, 2019). Another interesting concept using radar is, for example, presented by Cornick et al. (2015). Their approach relies on a ground-penetrating radar that is mounted under the vehicle and used for detecting features below the road surface. In the context of SLAM applications, Dissanayake et al. (2001) showed that radar-based point-landmarks are a promising concept for vehicle localization. More recently, Schuster et al. (2016) presented a radar-based localization that also integrates wheel-tick and IMU data. Their approach is to extract features from the vehicle's road environment in an accumulated radar measurement grid, which is afterward matched to a self-generated radar landmark map. The above approaches highlight that radar sensors can successfully contribute to the robustness of localization approaches, which is why we consider radar-based landmarks as one of the inputs for our approach. Closely related to this thesis is the work of Jürgens et al. (2020). They present an approach for radar-based odometry estimation and radar landmark extraction, which are used as input data for the graph-based sliding window approach presented in this thesis. Their work emphasizes the general-purpose characteristics of our approach and shows that it is applicable on a purely radar-based system.

Within autonomous driving applications, LiDAR sensors are one of the predominant choices for localization. The main advantages of LiDAR sensors are their precise spatial resolution, comparably low sensor noise, and wide detection range. A LiDAR-based localization approach is presented by Levinson et al. (2007), who use LiDAR reflectivities that are matched to a 2D dense reflectivity map. Their approach integrates GPS, IMU, and wheel odometry data. Akai et al. (2017) present an approach that uses voxelized LiDAR raw data and a prerecorded LiDAR voxel map for localization on mountainous public roads. To be more robust against environmental changes, they extend their approach to incorporate road markings, which are detected through LiDAR-based ground point reflectivities. Wolcott and Eustice (2015) investigate a 3D LiDAR-based scan matching approach for localization to cope with challenging weather conditions by utilizing grid-based height distributions. In comparison, the approach by Baldwin and

Newman (2012) uses a horizontally and a vertically mounted 2D LiDAR in conjunction with a previously recorded 3D point cloud map. Serafin et al. (2016) tackle the challenge of matching sparse LiDAR point clouds to dense point cloud maps by using extracted line and plane like features based on principle component analysis. The approach by Wang et al. (2017) uses curb landmarks detected with LiDAR scans for localization in urban environments. Complementary to this thesis, we investigated the use of non-semantic LiDAR features for localization on a pre-generated LiDAR feature map (Hungar et al., 2020). The applied graph-based sliding window approach is based on the framework introduced in this thesis, whereas the LiDAR feature extraction is presented by Hungar et al. (2019). The presented results suggest that further work on the feature extraction is required to improve the achieved accuracy such that the presented feature-based approach is applicable to automated driving. Brenner (2009) proposes to use pole landmarks extracted from LiDAR for global localization, which is applied to a system with an automotive-grade LiDAR by Schlichting and Brenner (2014). Schaefer et al. (2019) also use the pole concept based on LiDAR for urban localization and show that it works on 15-month-old maps. Within this thesis, we use the concept of LiDAR poles as one of the landmark types for the evaluation of our presented localization approach.

Multi-sensor fusion for localization and mapping is applied in many different domains and platforms, e.g., in unmanned aerial vehicles (Klingbeil et al., 2014), unmanned underwater vehicles (Eustice et al., 2006), mobile ground robotics (Newman et al., 2009), and others. A cross-modality approach for road vehicle localization is presented by Sefati et al. (2017), who compare pole detections based on stereo depth images with poles detected from LiDAR data. They conclude that LiDAR-based poles yield more robust results than camera-based poles. Schindler (2013) approaches the localization problem by fusing camera-based lane marking and LiDAR-based pole detections in rural scenarios. A cross-modality approach for urban localization is presented by Wolcott and Eustice (2014), who use a LiDAR for map generation and a mono camera for localization. They infer the vehicle pose by comparing live images to multiple generated synthetic images based on a 3D LiDAR map. Another approach that combines multiple landmark types is presented by Lundgren et al. (2014). They fuse camera-based lane markings as well as unclassified radar landmarks, which include guardrails, reflector posts, and traffic signs, with GPS, wheel speed, and gyroscope data. In contrast to using individual detectors for each sensor modality, Deusch et al. (2014) present an approach that uses the same feature detector for a front and a rear-facing camera as well as a LiDAR-based grid map. Kümmerle et al. (2019) present an urban localization approach that combines LiDAR-based building facades and pole landmarks, as well as camera-based road markings. Wu et al. (2017) present

a similar approach that combines camera-based lane markings and uses blob features extracted from a 2D LiDAR-based occupancy grid.

In this thesis, we present a generic localization framework that allows incorporating landmarks independent from the applied sensors as long as the measurements are given in the vehicle reference frame (VRF). In the scope of our evaluation, we discuss landmarks from radar and demonstrate our approach based on landmarks from LiDAR and camera.

2.4 State estimation for localization

Besides sensor technologies and landmark types, another fundamental aspect of perception-based localization approaches is the underlying state estimation technique. Our work is closely related to the SLAM domain, in which map-based localization can be seen as a subcategory. Over the last 30 years, extensive research has been done in the field of mapping and localization. An introduction to the topic is presented by Durrant-Whyte and Bailey (2006) and Bailey and Durrant-Whyte (2006). We refer to Cadena et al. (2016), who provide a broad overview of the challenges and open questions in the field. Similarly, the work of Huang and Dissanayake (2016) reviews theoretical aspects of SLAM approaches. A recent overview with a particular focus on autonomous driving is given by Bresson et al. (2017). For a broader overview of technologies applied within automated driving and their relation to localization, we refer to Yurtsever et al. (2020). The state estimation techniques applied for pure map-based localization are often either designed as a standalone approach or based on a closely related SLAM approach. Predominantly, the methods of choice are either particle filters, Kalman filters, or graph-based optimization. Stachniss et al. (2016) review and compare these three methods and provide a taxonomy for categorizing SLAM-related problems. Less frequently but still employed are histogram filters as a Bayesian method, e.g., Fox et al. (1999) and Bârsan et al. (2018). A disadvantage of those approaches is that they require a discretization of the state space, which can be a limiting factor for applications that require continuous poses at a high frequency. Although deep learning techniques for global localization, like Valada et al. (2018) or Naseer and Burgard (2017), are recently gaining more attention, they are still drastically outperformed by more classical techniques. Nowadays, graph-based optimization is considered the de-facto standard for SLAM applications. Some of the fundamental works that lead to formulating SLAM as a graph-based optimization problem are Lu and Milios (1997), Frese and Hirzinger (2001), Folkesson and Christensen (2004), and Thrun and Montemerlo (2006). Graph-based approaches for map-based localization only recently gained attention in the last few years. From a mathematical perspective, our localization approach is closely re-

lated to the non-linear least squares formulation of graph-based SLAM presented by Grisetti et al. (2010). Another option for formulating graph-based optimization problems is to view it as a probability maximization problem. We show the connection between probability maximization and least squares error minimization in Section 3.4.1. The work of Agarwal et al. (2014a) connects graph-based SLAM approaches to techniques known from the Geodesy domain. They identify similarities and point out that the underlying problems in both domains are often related. For an overview on the Geodesy domain's mathematical perspectives, we refer to Niemeier (2008).

The approach by Levinson et al. (2007) uses graph-based SLAM for a dedicated map generation, while the localization is carried out with a particle filter. Yielding higher precision estimates, they enhance their approach in later work to use a histogram filter instead of a particle filter (Levinson and Thrun, 2010). A more recent approach that relies on histogram filtering is presented by Ma et al. (2019). They first correlate landmark detections with a map and afterward perform sensor fusion through an additional correlation step in a Bayesian fashion for highway driving. In contrast, Schuster et al. (2016) apply graph-based SLAM for mapping but use random sample consensus (RANSAC) map matching for pose estimation. The approach by Spangenberg et al. (2016) first employs a particle filter for map-based localization and additionally filters the resulting pose estimates with a subsequent Kalman filter. In contrast, Ziegler et al. (2014) only rely on Kalman filters. The localization approach by Engel et al. (2019) combines a deep learning approach for odometry estimation and relies on a Kalman filter for integrating temporal information. Another combination is demonstrated by Wolcott and Eustice (2014), who use pose graph SLAM for mapping and an Extended Kalman filter for localization. Similarly, Poggenhans et al. (2018) present an approach that uses a Kalman filter for localization and relies on pose graph optimization in their mapping pipeline. In the work of Schreiber et al. (2013) the mapping process does not require a dedicated state estimation technique because it is only based on highly precise GNSS data and manual labeling. For localization, they employ a Kalman filter. Similarly, Aeberhard et al. (2015) use a Kalman filter for pose estimation based on landmarks and GPS data. The approaches by Lundgren et al. (2014), Deusch et al. (2014), and Schindler (2013) are examples of various variants of using particle filters for localization. Their main difference is the sensor setup, landmark types, and particle weighting. More recent particle filter approaches for urban vehicle localization are Sefati et al. (2017) and Schaefer et al. (2019), which show that particle filters remain popular. This is also underlined with the approach presented by Jonschkowski et al. (2018), which is a particle filter formulation that allows for end-to-end learning of measurement and motion models. We refer to the work of Stachniss and Burgard

(2014), who provide an overview of various approaches to localization and SLAM using particle filters. Within the evaluation of this thesis, we use the particle filter implementation presented by Stess (2017) and compare it to our graph-based optimization approach.

2.4.1 Robust estimation

Handling outliers is an essential aspect of robust optimization techniques. Due to the quadratic weighting of error terms in non-linear least squares optimization, outliers have a drastic effect on the optimization result. A common approach for robustification is to alleviate outliers' influence by downweighting their large error terms during optimization. The idea is to modify the original optimization problem by applying a robust cost function. In statistics literature robust optimization is referred to as M-Estimation (Huber and Ronchetti, 2009). We present this technique in more detail in Section 3.4.4.

As robust optimization is a well-studied field, several options for robust cost functions have been proposed. Classical choices are the Huber kernel (Huber, 1964), Welsch kernel (Dennis, Jr. and Welsch, 1978), Geman-McClure kernel (Geman and McClure, 1985), and Cauchy kernel (Black and Anandan, 1996). The latter is also known as the Lorentzian kernel. Which robust cost function is applicable depends on the nature of the dataset and the characteristics of the error function that is robustified.

An essential aspect of graph-based localization is solving the data association of landmark measurements. This may include solving loop closures in SLAM applications, associating landmarks to a map, and finding out if multiple landmark measurements belong to the same object. The data association problem is naturally ambiguous if multiple similar landmark configurations in the real world exist. Within the SLAM domain, this is also known as *perceptual aliasing*. It leads to incorrect data associations and thus might cause localization failure. Therefore dealing with outliers has been widely studied in the SLAM domain.

Agamennoni et al. (2015) investigate a concept, called self-tuning M-estimators, for choosing a robust cost function based on the likelihood of the data given the best fitting distribution model. They interpret M-estimators as being derived from elliptical distributions, which helps tuning the M-estimator's parameter automatically.

Barron (2019) presents a generalized robust cost function that represents a broader family of well known traditional cost functions. The presented cost function can be adjusted by tuning a single shape parameter, which allows deriving, e.g., L2, Cauchy, German-McClure, Welsch, and other cost functions. Besides, Barron proposes an algorithm that automatically tunes the shape parameter during optimization without the need to set any hyperparameters. In return, the

optimization problem is adaptively robustified in an ideal way. Chebrolu et al. (2021) enhance this method by proposing an algorithm that iteratively optimizes the shape parameter and cost function based on an Expectation-Maximization approach instead of jointly optimizing both.

Olson and Agarwal (2013) present an approach called Max-Mixture (MM) that models multi-modal beliefs through a sum of Gaussians. Instead of incorporating all modes of the sum of Gaussians, they suggest selecting the component with the maximum log-likelihood during optimization, which effectively deactivates erroneous modes of the Gaussian. Their approach is applied for eliminating erroneous loop closure constraints by modeling the loop closure and a null hypothesis in a two-component Gaussian. Additionally, their approach is demonstrated for odometry outlier rejection by modeling separate modes for slip and grip measurements.

Latif et al. (2013) introduce the Realizing, Reversing, Recovering (RRR) approach that clusters loop closure constraints based on similarity and extracts the largest subset of consistent clusters. RRR requires that the optimization successfully converges in order to reveal erroneous constraints.

Sünderhauf and Protzel (2012) present an approach called Switchable Constraints (SC), in which the topology of a graph itself is optimized. They augment loop closure constraints with a switching variable which are controlled through switching priors to prevent that all constraints are deactivated. More theoretical aspects are covered by Sünderhauf (2012). Lajoie et al. (2019) present an approach that improves upon SC by using truncated least squares and show that modeling the correlation between outliers also improves the performance. Similarly, the approaches by Yang et al. (2020b) and Yang et al. (2020a) use truncated least squares formulations to model outliers within point cloud registration and shape alignment tasks. In addition to previous work, both approaches also provide criteria for certifying results. Agarwal et al. (2013) propose an improvement to SC called dynamic covariance scaling (DCS). They present a closed-form solution that is applied as a weight to the covariance matrix of each constraint. The weight dynamically scales the covariance based on the original quadratic error term. Their approach is popularly used as it is easily integrated into commonly used optimization frameworks. As experimentally shown by Agarwal et al. (2014b), DCS improves SLAM applications under poor initial estimates and handles data association outliers.

MacTavish and Barfoot (2015) compare various choices for robust cost functions. They find that DCS and Geman-McClure are well suited for the task of treating correspondence outliers in visual feature-based applications. Both cost functions behave similarly for larger errors, while DCS is equivalent to standard least squares for smaller errors. Sünderhauf and Protzel (2013) compare RRR,

MM, and SC on different datasets for pose graph SLAM. They point out that all three methods have varying performance, and there is no clear winner in their comparison. A further comparison of RRR, MM, SC, and DCS is presented by Latif et al. (2014). In addition to a benchmark-based comparison, they also provide an argumentative comparison. They find that DCS is the fastest but also highlight that SC and DCS could trap the optimization in local optima.

2.4.2 Sliding window graphs

For automated driving, any localization algorithm must be computationally efficient as the car requires pose estimates in (near) real-time. Within graph-based approaches, one way of achieving computational tractability is to limit the optimization problem's size by using sliding windows over states. In general, optimizing over a set of consecutive states, i.e., a temporal sliding window, is known as *smoothing*. An advantage of smoothing techniques is that the states within the sliding window can be used for relinearization during optimization (Julier, 2003; Dellaert and Kaess, 2006). In contrast, *filtering* techniques like Extended Kalman Filter (EKF) only update states at a single timestamp such that linearization errors accumulate over time, which might lead to inconsistent results (Julier and Uhlmann, 2001). Related to this, Strasdat et al. (2012) demonstrate for visual SLAM that keyframe bundle adjustment, which is similar to sliding window methods, is superior to filtering. More close to our approach is the work of Sibley (2006), who highlights the benefits of sliding window filters coupled with delayed marginalization of poses and landmarks. The approach is also demonstrated by Sibley et al. (2010) for planetary landing applications and used by Newman et al. (2009) for urban SLAM based on a modified Segway platform. In comparison, we explicitly consider the choice of linearization points for further reducing linearization errors and present a sparsification scheme for sliding window graphs that prevents marginalization fill-in. Another beneficial aspect of sliding window graphs is presented by Ranganathan et al. (2007). They introduce an algorithm for pose estimation based on fixed-lag smoothing and emphasize the benefits of integrating out-of-sequence measurements within sliding window graphs.

We explain the sliding window concept used in this thesis in more detail in Section 4.2 and cover related work for road vehicle localization in the following. Lategahn et al. (2013) present an approach that uses graph-based optimization for map creation and also for localization. Their localization is formulated as a two-step sliding window optimization problem. In each optimization cycle, they first detect visual landmarks in an image and optimize a single pose estimate such that the back-projection error is minimized. These pose estimates are then used as constraints in a sliding window graph that further integrates IMU measurements.

Similarly, the work of Chiu et al. (2013) uses a sliding window graph framework that is divided into short-term and long-term smoothing. In their approach, both smoothers share a common map. While the long-term smoother integrates loop closure constraints and is optimized at a low frequency, the short-term smoother is optimized at a higher frequency to maintain constant-time localization updates. Wu et al. (2017) investigate a localization approach based on pose graph optimization. In comparison to their work, our graph-based approach benefits from a different data association strategy and grid-independent landmark detections. Harr et al. (2018) present a pose graph optimization approach to localization that is closer to our approach. Compared to our work, we use a different data association strategy and incorporate landmarks into state optimization, which results in a more accurate system. The recently presented approach by Kümmerle et al. (2019) uses pose graph optimization and integrates landmarks in a local map similar to our approach. The difference to our approach is that we do not rely on a pose graph but instead integrate point landmarks into the state vector, which allows us to explicitly constrain map-matched landmarks through position priors.

The sliding window graph formulation in this thesis is closely related to the work of Merfels and Stachniss (2017). They present how to synchronize odometry and pose measurements in sliding window pose graphs. We extend their framework in this thesis and show how to integrate landmarks in the sliding window graph formulation. We implement our sliding window graph approach within the g2o framework presented by Kümmerle et al. (2011a). A similarly useful framework that eases implementation effort and shares a similar nomenclature as used in this thesis is called GTSAM and presented by Dellaert (2012). Another popular choice is the Ceres Solver by Agarwal et al. (2020). In contrast to the mentioned related approaches, we incorporate a given third-party map as unary priors for landmark states in our state estimation. Instead of directly changing the map, we use the estimated landmark positions for delayed map refinement through a back-end service. Another aspect that distinguishes graph-based sliding window approaches is how they handle measurements that drop out of the sliding window. We cover the different options separately in the following section.

2.4.3 Marginalization and sparsification

Compared to full graph solutions, marginalization allows preserving information while limiting the state dimension. The latter is necessary to ensure computational tractability in online scenarios. The common drawbacks of marginalization are the missing option of relinearization and induced fill-in in the graph and its corresponding system matrix. The latter one negatively influences the sparsity pattern and thus increases computation costs. For the case of pose graphs, Merfels and Stachniss (2016) presented in their work how to compute pose priors that

approximate pose graph marginalization.

One of the options for reducing the number of constraints induced by marginalization is commonly called sparsification. The Sparse Extended Information Filter (SEIF) by Thrun et al. (2005) selectively deactivates constraints between the current robot pose and landmarks but keeps intra-landmark constraints. Eustice et al. (2005) extend the SEIF approach with a modified sparsification step that preserves the state mean and produces similar results as the full-covariance EKF. The work of Frese (2005) compares representing the SLAM problem in covariance form against representing it in information form. He shows that the information matrix representation is approximately sparse, and distant landmarks can be conservatively eliminated by removing off-diagonal entries. This is one of the foundations in the development of state-of-art graph-based SLAM techniques.

Vial et al. (2011) present a technique for conservative graph sparsification via Kullback-Leibler Divergence (KL) minimization. Huang et al. (2013) introduce a sparsification scheme that first sparsifies nodes through marginalization while retaining consistent relative constraints. Afterward, they sparsify edges by solving a l_1-regularized minimization problem. Kretzschmar et al. (2011) use a Chow-Liu tree to approximate marginalization and maintain a sparse graph. The idea of Chow-Liu trees is also used in the context of appearance-based localization within the FAB-MAP algorithm, Cummins and Newman (2007) and Cummins and Newman (2008). The idea of Kretzschmar et al. (2011) is extended by Carlevaris-Bianco and Eustice (2013), Carlevaris-Bianco et al. (2014), and Carlevaris-Bianco and Eustice (2014) through introducing a conservative tree approximation. Using more complex topologies than trees as an optimization technique for sparsification is suggested by Vallvé et al. (2017), Vallvé et al. (2018), and Vallvé et al. (2019). Without explicit marginalization, Choudhary et al. (2015) reduce poses and landmarks based on expected information gain. Ta et al. (2018) present a near-optimal method for pose marginalization by reparameterizing to local spaces.

More closely related to our work, Hsiung et al. (2018) investigate how to approximate marginalization with sparse priors. While their work focuses on localization errors in the visual-inertial odometry domain, we focus on estimating landmark positions for automated driving. Additionally, they do not cover the effects of different linearization points and rely on suboptimal global linearization points. Eckenhoff et al. (2016) provide a derivation that suggests using local linearization points for marginalization but requires local optimization of the marginalization blanket. Mazuran et al. (2014) show how to recover artificial measurements from marginalized information, which allows for relinearization. They further argue that local linearization points provide superior results in incremental mapping scenarios. An extended version of their work is Mazuran et al.

(2016).

Compared to the related work, we show how to avoid using local linearization points and derive how to utilize global linearization points by respecting gradient effects inside our sparse priors. We study the effects in relation to map refinement.

2.5 Maps for road vehicle localization

In the following, we distinguish between localization approaches that rely on *tailored maps* and approaches that use *general-purpose maps*. We conclude this section by discussing approaches for landmark map refinement.

2.5.1 Tailored maps

The category *tailored maps* includes localization approaches that operate on a map that was created with a sensor setup that is similar to the one used for localization. This is usually the case for approaches that share a common methodology between map creation and localization, e.g., in SLAM techniques. Likewise, approaches with a methodologically different mapping process also belong to this category as long as they operate on a similar sensor setup as it is used for localization. An advantage of using the same setup for mapping and localization is that the map reflects and matches the vehicle's sensor characteristics. Therefore, the map only contains elements that the vehicle can detect and no additional elements. In return, the underlying map matching is easier as it must not consider systematic differences between the map's content and the detectable elements. In general, a significant disadvantage of such approaches is that it is often unclear if the created maps are transferable to other sensor setups. An implication is that maps might need to be recreated if sensor setups are changed. This limits the applicability to series production, especially if the covered map areas are huge. Many localization approaches rely on previously built and tailored maps. We mention some of them in the following for further reference. An example of tailored maps are LiDAR-based 2D ground reflectivity maps proposed in the work of Levinson et al. (2007) and Levinson and Thrun (2010). In comparison, the approach by Schuster et al. (2016) requires a previously built 2D radar feature map. A popular choice is to rely on point cloud maps, for example, Baldwin and Newman (2012) and Withers and Newman (2017). A broad share of literature in the domain focuses explicitly on generating point cloud maps. For example, Yang et al. (2018) present how to generate 3D point cloud maps at a city scale. One of the main challenges in using maps for localization is to ensure long-term localizability. The approach by Egger et al. (2018) uses LiDAR-based voxel features in conjunction with multiple local sub-maps for localization in changing environ-

ments. They achieve long-term localization through individually updating local sub-maps if deviations are detected. In the context of deep learning assisted localization, Wei et al. (2019) present an approach for compressing highly accurate LiDAR intensity images used as a map. Their approach aims to reduce the required storage for their LiDAR-based intensity map, which is one of the main disadvantages with respect to scaling them. All of the above approaches are tailored towards specific sensors, limiting their transferability to other sensor setups. In comparison, we review approaches that use general-purpose maps in the following.

2.5.2 General-purpose maps

The category *general-purpose maps* refers to maps that are used for multiple applications and purposes. A property of general-purpose maps usually is that they represent the environment in a generic way such that the elements of a map have a specific semantic meaning. The challenge of using these maps for localization is that the map elements must not necessarily correspond to what the vehicle can detect. For example, the approach presented by Vysotska and Stachniss (2016a) matches LiDAR data to OpenStreetMap (OSM) data for localization. They explain their approach in more detail in subsequent work, see Vysotska and Stachniss (2017). Other approaches that utilize OSM data for localization are, e.g., Floros et al. (2013), Landsiedel and Wollherr (2017), Ma et al. (2017), Fleischmann et al. (2017), and Brubaker et al. (2016). All of these approaches show that OSM data can be used to improve the accuracy of localization estimates. Nevertheless, so far none of the localization approaches using OSM or other comparable maps are accurate enough for automated driving. Agarwal et al. (2015) present an approach using visual feature points extracted from monocular images matched to Google Street View panoramas for localization. They show that their approach is suitable for localizing a smartphone in urban environments. However, due to the limited accuracy, their approach does not seem promising for automated driving applications. The approach by Kümmerle et al. (2009) and Kümmerle et al. (2011b) improves localization estimates by finding correspondences between LiDAR scans and aerial images, which in a way can be seen as a map. Similarly, Tang et al. (2020a) present how to use radar data in combination with aerial images for localization. They also extend their approach to work with LiDAR data (Tang et al., 2020b). Another type of map is used by Lee et al. (2007) , who suggest incorporating road network maps, e.g., navigation maps, as priors for their 2D LiDAR-based localization system. Their approach effectively restricts pose estimates to be close to the road. We consider this approach as potentially hazardous as vehicle poses that, in reality, are next to the road might be estimated as on the road. Roh et al. (2016) present an approach that leverages

available geospatial information, like digital elevation models and building planes as polygon chains. Their approach is mainly focused on map creation rather than localization. Nevertheless, various aspects of their work can be transferred to pure localization applications. Although all of the above approaches show that they are eligible for improving localization and mapping applications based on existing general-purpose maps, the achieved accuracies also highlight that highly accurate map-based localization also requires highly accurate maps.

In general, approaches that use dense 3D point clouds as maps but use other sensors for localization can also be included in the general-purpose maps category. For example, the approach of Caselitz et al. (2016) matches visual features to a given 3D LiDAR point cloud map for map-relative localization. Chen et al. (2019) present a SLAM approach for generating LiDAR-based surfel maps that contains a semantic class for each surfel. Their work highlights that semantic meaning in maps is beneficial for localization. However, a significant disadvantage of using dense 3D point cloud maps is that they require an extensive amount of memory if used in large-scale applications, which is why we do not further consider them.

More close to our work is the use of a third-party map, as shown by Wu et al. (2017). Their map is designed for automated driving use cases and includes globally accurate landmarks. Their landmark detection is based on building occupancy grids and blob feature extraction to extract distinct landmarks. Their map matching is a nearest neighbor search, and data associations are solved within graph optimization. In comparison, our approach does not rely on grid discretization, follows a different map matching technique, and solves data association before optimization. Although their sensor setup and dataset is different from ours, comparing the results suggest that our method is more reliable and accurate. The approach by Poggenhans et al. (2018) also utilizes a highly precise map for automated driving that contains road markings and lane boundaries. Their approach uses an Unscented Kalman Filter for localization and is demonstrated for camera-based localization in an urban scenario. The limited availability of their approach on narrow urban road highlights the importance of fusing different landmark types for localization.

Bresson et al. (2017) state that localization approaches based on existing maps are not yet suitable for automated driving. The prevailing reasons are the lack of accuracy in large-scale maps, and that accurate maps are only available for limited areas. Mapping companies are nowadays working on providing highly accurate maps for large areas, which could close this supply gap soon. Therefore, we investigate how to use such a general-purpose third-party landmark map as prior knowledge for localization in this thesis.

2.5.3 Refining landmark maps

A fundamental challenge to all approaches relying on pre-built maps is that the maps are getting outdated over time. In automated driving, temporary construction sides and newly built streets directly impact the map's correctness and thus localization. Pauls et al. (2018) present a case study investigating the road changes on a German highway that render a map outdated. They distinguish between major road changes that invalidate entire parts of the map and minor changes that do not have a safety-critical impact on automated driving functions. The latter describes the map changes that we want to estimate within this thesis and thus prevent that minor changes aggregate and become safety-critical over time. Pannen et al. (2020) point out that it seems unsuitable for large scale applications to rely on specialized mapping vehicles for capturing map updates. They argue that the frequency of traversals would be insufficient and propose to use camera data from nowadays series vehicles for mapping highways. Similarly, approaches like Skibinski (2019) and Sons and Stiller (2018) tackle updating maps for automated driving through fusing collective vehicle data on a back-end server but focus more on rural and urban areas. Disadvantages of these methods are the possibly extensive amount of data that needs to be transmitted and the delay until updates are received by the vehicle.

Narula et al. (2018) investigate the accuracy limit of GNSS-based mapping achievable with low-cost receivers through collective data. They state that sub-50 cm accuracy is feasible. As this is not sufficient for automated driving, it is necessary to aid mapping through RTK-based GNSS or large-scale SLAM-based methods to achieve sufficient global accuracy. An example of such a dedicated system for mapping based on point clouds is presented by Yang et al. (2018). These highly accurate point clouds are commonly either directly used for localization or the foundation for creating and updating landmark maps. A further aspect in the context of updating landmark maps is the concept of using temporal decay for map elements. Skibinski et al. (2016) suggest using temporal weights that decay over time and are reset whenever a landmark is measured. This allows a back-end server to dismiss landmarks that have been removed. A similar idea is presented by Rosen et al. (2016), who show how to measure the temporal persistence of features. They show how to model periodic disappearance or other temporal patterns and model the expected survival time for individual features. The approach is successfully applied by Chebrolu et al. (2019) in the domain of updating crop field maps. Their approach is from a methodical point not far from updating landmark maps for automated driving applications and therefore is also promising for urban scenarios.

Within this thesis, we focus on estimating landmark update hypotheses directly in the vehicle. We conceptually discuss sending the update hypotheses

over-the-air to a back-end service such that validated map updates can be computed based on fleet data. We extend our graph-based sliding window approach to localization in Chapter 5 w.r.t. refining landmark maps. A key factor within our extension is how we deal with measurements that drop out of the sliding window. We use delayed marginalization similar to Sibley et al. (2010) but show how to apply a further sparsification step to preserve our sliding window graph's sparsity. In addition, we explicitly consider the choice of linearization points for further reducing linearizations errors.

Chapter 3

Fundamentals

In this chapter, we explain the fundamentals required for the rest of this thesis. We explain the basics that are relevant throughout the whole thesis and introduce fundamental algorithms and concepts that we use within this thesis.

3.1 Coordinate systems

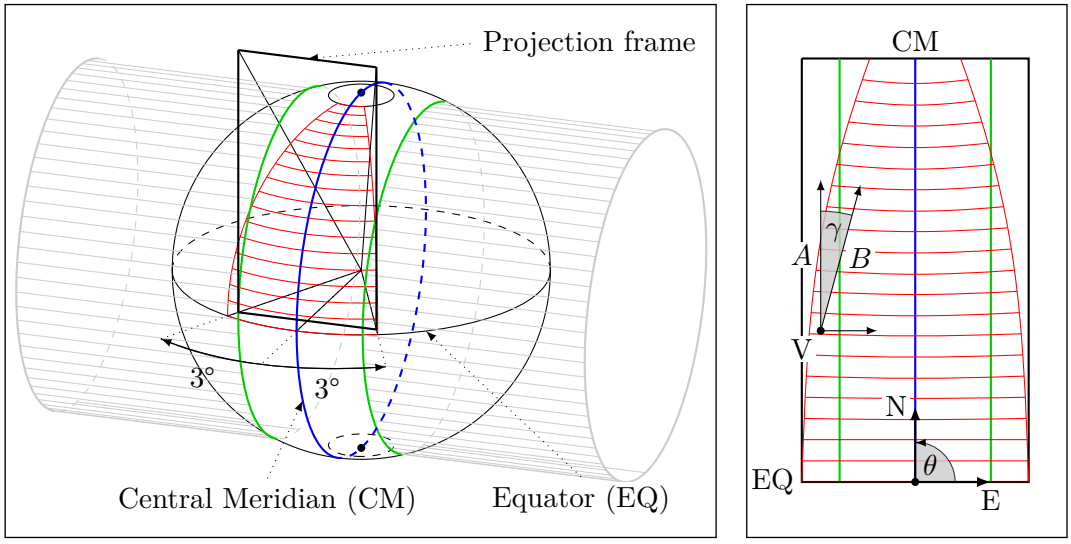
In the following, we explain the world reference frame (WRF) and vehicle reference frame (VRF) as used in this thesis and discuss the consequences of calibration errors when projecting landmark measurements into the VRF.

3.1.1 World reference frame

We choose to represent global information like vehicle poses and map landmarks in the Universal Transverse Mercator (UTM) coordinate system, which is widely used across various domains and a popular mapping standard. The main advantage of using UTM is that it is based on representing the world in Cartesian coordinates, which makes it straightforward to compute Euclidean distances and transformations. The UTM system divides the globe into 6° wide segments, yielding in total 60 different UTM zones. Each zone is defined by its central meridian and individually projected onto a perpendicular secant cylinder. By design, the projection cylinders are scaled down by a factor of 0.9996 compared to its defining central meridian, which balances the scale variation within a segment. The cutting circles in which the cylinder intersects the globe represent the lines of true scale. The scale is shrunk between the two cutting circles and expanded outside, which minimizes the overall scale variation. Unfolding the cylinder then produces a Cartesian coordinate system in which the central meridian defines the x-axis (Easting) and the equator corresponds to the y-axis (Northing). The origin at the equator's intersection with its central meridian of each UTM zone is set to $(500\,000\,\mathrm{m}, 0\,\mathrm{m})$ for the northern hemisphere and to $(500\,000\,\mathrm{m}, 10\,000\,000\,\mathrm{m})$ for

the southern hemisphere, which prevents negative coordinates. A more detailed description of the secant transverse Mercator projection and the UTM system definition is provided by Snyder (1987). We illustrate the secant transverse Mercator projection and the derived UTM coordinate system in Figure 3.1. The vehicle orientation in each UTM zone is zero if the vehicle faces east, and it increases counter-clockwise, which is shown in Figure 3.1b. Also, Figure 3.1b illustrates the meridian convergence, which is the deviation between true north and grid north. The deviation increases with growing distance to the central meridian. It must be considered when converting the vehicle's position and orientation to, e.g., the World Geodetic System (WGS) coordinate system.

Concerning the methods presented in this thesis, the UTM system could be substituted with any other Cartesian coordinate system without implications on the presented approaches. This might especially be relevant for, e.g., localization within parking garages or other locally defined maps.



(a) Secant transverse Mercator projection

(b) UTM Coordinates

Figure 3.1: Universal Transverse Mercator (UTM) projection. Both figures illustrate how the coordinate system of a UTM zone is constructed. For illustrative purposes, the proportions are heavily exaggerated, and only the northern hemisphere is shown in detail. Projections of the southern hemisphere are made accordingly. (a) A UTM zone with its central meridian and its in total 6° wide segment, which is shown in red. The segment is projected onto the secant cylinder, which is unfolded to produce the projection frame. The green circles denote the cutting circles between the globe and cylinder. (b) The origin of the UTM zone coordinate system is defined at the intersection between the equator (EQ) and central meridian (CM). Its x-axis is the Easting (E) direction, whereas the y-axis is the Northing direction (N). If a vehicle at position V is oriented towards grid north, i.e., in the direction of vector A, true north deviates, which is in the direction of B, by the angle γ.

3.1.2 Vehicle reference frame

The VRF used in this thesis is similar to the ISO 885:2011 definition by the International Organization for Standardization (2011). The coordinate system's origin is defined in the ground projected middle of the vehicle's rear axle. Its x-axis points towards the front of the vehicle, whereas the y-axis direction is to the left side of the vehicle. The vehicle orientation starts at zero within the x-axis and increases counter-clockwise. Figure 3.2 illustrates the reference system. Within the scope of this thesis, we use the VRF to describe the vehicle's relative movement within a specific time interval. These are called odometry measurements. Also, we expect landmark measurements to be available in the VRF. This allows us to employ a generic interface between landmark detectors and our localization approach. In contrast, each sensor modality could supply landmark measurements in its sensor reference frame (SRF), which would require that each sensor must be individually incorporated into our approach. Instead, we abstract from the specific sensor modality and use the VRF as an abstraction layer. This allows us to interchangeably incorporate landmark measurements from LiDAR, radar, camera, and other sensors without adapting our approach.

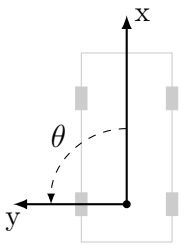
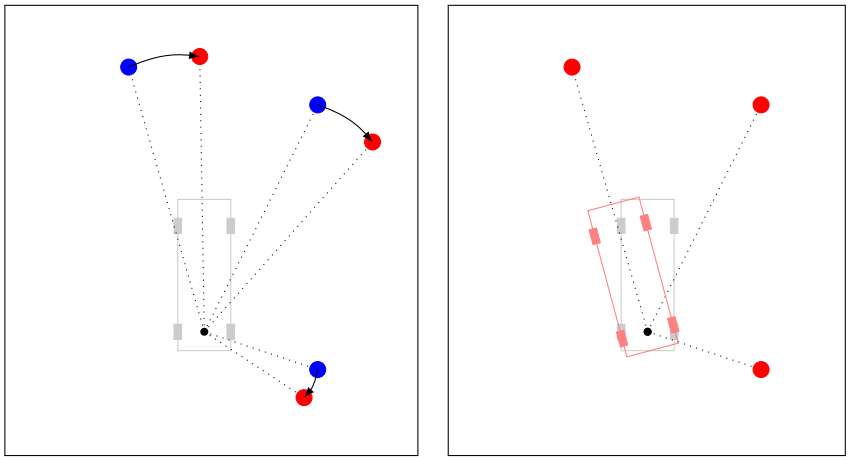


Figure 3.2: Vehicle reference frame as used in this thesis. The y-axis increases to the left of the vehicle, whereas the x-axis increases towards the front of the vehicle. The vehicle's orientation θ is zero within the x-axis and increases counter-clockwise.

3.1.3 Importance of calibration

We assume that landmark measurements are available to our localization approach within the VRF. Therefore, landmark detectors must either perform landmark detection directly in the SRF and afterward transform the detected landmarks into VRF or first transform measurement data into VRF and afterward perform landmark detection. In both cases, it is crucial to have a valid and accurate transformation matrix from SRF to VRF. In the context of this thesis, we understand estimating such a transformation matrix as sensor calibration. While i.i.d. noise of landmark measurements is acceptable and can be handled within our optimization approach, systematic calibration errors have a direct impact on

our localization approach. Although the effects on the estimated vehicle pose are possibly mitigated by taking into account multiple sensor modalities, the negative impact on estimating the position of a landmark remains if it is only measured with a single sensor. An angular calibration error of 0.5° already displaces a landmark measurement that was taken at a 100 m distance by approximately 0.9 m from its true position. An example of a calibration error with a constant angular offset is illustrated in Figure 3.3. For the sake of convenience, we assume that sensor calibration is solved upfront and independent of our approach. Although various methods for online calibration exist, e.g., Levinson and Thrun (2013) and Mirzaei et al. (2012), we assume in this thesis that a static calibration is given.



(a) (b)

Figure 3.3: Exemplary impact of an angular calibration error between SRF and VRF. (a) Instead of measuring the true landmark positions in blue, the erroneously miscalibrated measurements in red are produced. (b) As a consequence, the vehicle's orientation is falsely rotated when matching the measurements to their true position. Within our sliding window approach, the effect on the vehicle pose is mitigated by fusing the measurements of several sensor modalities over a specific time interval.

3.2 Landmarks

So far, we mentioned that we rely on landmark measurements that are associated with a map in order to compute a global vehicle pose. In this section, we explain and define in more detail what we understand as a landmark. Afterward, we present the landmark types that we incorporate into the localization estimate.

In literature, the term feature is often used exchangeable to the term landmark, whereas we distinguish between both of them in this thesis. A feature (also referred to as a key point) is in computer vision a pixel of interest in an image. It describes, for example, a significant change in brightness or color. Sim-

ilarly, the term is also used for other sensors like radar and LiDAR. In the case of LiDAR sensors, a feature point could be a sudden jump in depth or reflectivity. While these examples only give a brief idea about the nature of a feature, its clear definition usually depends on the sensor being used and the application. Defining feature descriptors is an active and open research field. With respect to automated driving, one of the main challenges is to find feature descriptors that are stable and persistent over time and in varying weather conditions. In contrast to features, we understand landmarks as being more semantically distinct and human interpretable. The term landmark is also used in marine navigation, where it describes an object, like a church tower or lighthouse, that is easily visible from a distance. Thus, the position of the boat can be inferred by taking angular measurements of multiple landmarks and then performing triangulation. The same concepts also apply to celestial navigation, only with much larger distances and celestial bodies. From a conceptional point of view, these types of landmarks are already close to the ones applicable for automated driving. In this thesis, we rely on the general definition of Thrun et al. 2005, p. 21, and extend it for our use case.

Definition 1. *A landmark for automated driving is*

- *a stationary element of the environment,*
- *easy to recognize,*
- *reliably detectable with one or ideally multiple sensor modalities,*
- *visible from the road,*
- *has a preferably large viewing scope,*
- *might have additional properties that describe its appearance, and*
- *either a specific or abstract semantic type.*

Although our definition is strict in some points, it is intentionally interpretable in others. For example, in terms of semantic meaning, it would be optimal to distinguish between the trunk of a tree and a lamp post. The available detectors in the vehicle, however, might only be able to measure a cylindrical object without being able to distinguish both objects. Nevertheless, the common abstract description of a cylindrical pole can still be used as a landmark type. Similarly, additional properties like a radius or color of a pole are desirable and helpful for distinguishing measurements but not strictly necessary. The absolutely necessary thing that each landmark must fulfill is that it is stationary and must have a persistent and distinct location.

We distinguish in the 2D case between landmarks defined with a single position and landmarks that require multiple positions to define their location. We call the first one *point landmarks* and the second *polyline landmarks*. Throughout this thesis, we show how to utilize these landmarks for estimating the vehicle pose. An excellent example of a point landmark is a lamp post, whereas vegetation (e.g., a bush) is not suitable as its dimension and location might vary over time. Similarly, a fixed reflector post at the street side is more suitable as a landmark than an easily movable one used in construction sites.

Another important aspect is that landmarks for automated driving must be visible from the road. For example, a building usually is only visible from a few sides, whereas the backside is usually not measurable. This means a building itself is not a suitable landmark, whereas some of the building facades are. Besides, landmarks are ideally continuously visible from subsequent poses while a vehicle passes by. A landmark has a large viewing scope if it is visible from many locations. In return, a passing vehicle can measure it more often than a landmark with a small viewing scope. Therefore, landmarks with a larger viewing scope can be more beneficial for localization than landmarks with a comparably small viewing scope. It is crucial to consider the different aspects that a landmark must fulfill when designing a landmark-based localization system. Specific examples for landmarks used in this thesis are

- point landmarks based on lamp posts, reflector posts, traffic light posts, bridge pillars, and
- polyline landmarks based on curbs, building facades, solid lines, dashed lines, guardrails, and fences.

We show in Section 4.5 how to incorporate these landmark types into our graph-based localization system. Figure 3.4 gives an example of landmarks that qualify as suitable and unsuitable landmarks.

3.3 State definitions for vehicle localization

Throughout this thesis, we seek to optimize the vehicle's trajectory and the position of detected landmarks along with it. Based on our requirements for automated driving, we limit ourselves to estimating 2D poses and ground-projected landmarks. This is sufficient for our use case of automated driving in urban scenarios as our vehicle moves on a ground plane and map information is stored relative to the ground. The concepts of this thesis are, in general, also applicable to 3D poses and landmarks. In the following, we state our notation used in this thesis, which is essential for understanding our derivations and concepts in the

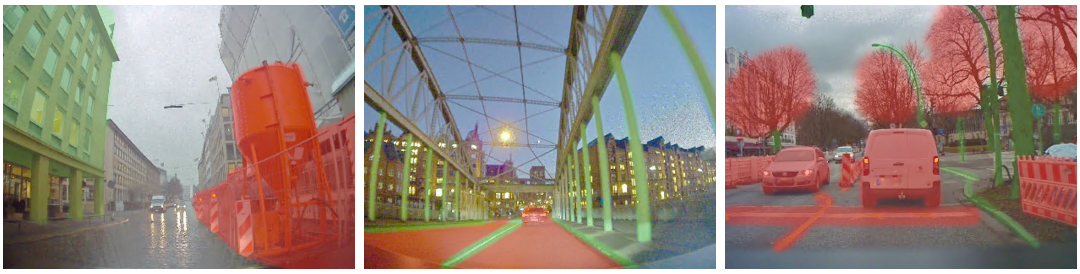


Figure 3.4: Exemplary scenes with suitable landmarks in green and elements that we consider unsuitable as landmarks in red. Left: The buildings and pillars on the left side are suitable as they are stationary. On the contrary, the construction site elements on the right side are not persistent and easily movable, thus unsuitable for long-term localization. Middle: Road markings, curbs, and bridge pillars are suitable landmarks as their position is definable as point or polyline landmark. Right: The stem of a tree is a suitable landmark, whereas tree crowns are unsuitable as they change their form and appearance through the rotation of seasons. Similarly, dynamic objects and construction site elements are unsuitable as they are temporary.

later chapters. We define the overall state vector that we want to estimate as the vector concatenation

$$\boldsymbol{x} = [\boldsymbol{x}^p, \boldsymbol{x}^l]^\top,$$

where $\boldsymbol{x}^p$ is the vehicle trajectory and $\boldsymbol{x}^l$ the vector of landmark states. In more detail, we represent the vehicle trajectory as

$$\boldsymbol{x}^p = [\boldsymbol{x}_1^p, \ldots, \boldsymbol{x}_N^p]^\top,$$

where each $\boldsymbol{x}_i^p$ is an individual pose at some timestamp, which is not further denoted in our notation, and the number of poses N. In our case, each vehicle pose is within a Lie group SE(2) such that

$$\boldsymbol{x}_i^p = [x_{\text{UTM}}, y_{\text{UTM}}, \theta_{\text{UTM}}]^\top,$$

where x_{UTM} is the UTM Easting coordinate, y_{UTM} the UTM Northing coordinate, and θ_{UTM} the vehicle's orientation w.r.t. the UTM grid zone. Similarly, we concatenate the individual landmarks that we want to estimate as

$$\boldsymbol{x}^l = [\boldsymbol{x}_1^l, \ldots, \boldsymbol{x}_M^l]^\top,$$

where $\boldsymbol{x}_i^l$ is a landmark that has been observed from one of the poses in the vehicle trajectory. Note that we do not necessarily include all landmarks that have been observed but carefully choose which ones we want to include in the state estimate. Table 3.1 summarizes the state definitions for vehicle poses and provides an overview of exemplary landmark state definitions.

Variable	Definition	Description
$\boldsymbol{x}^p$	$[x, y, \theta]$	a vehicle pose defined by position and orientation
$\boldsymbol{x}^l$		general landmark
$\boldsymbol{x}^{l_p}$	$[x, y]$	a point landmark, e.g., a pole
$\boldsymbol{x}^{l_d}$	$[x_1, y_1, x_2, y_2]$	a line element defined by two points
$\boldsymbol{x}^{l_c}$	$[x, y, \alpha, \theta]$	a corner defined by a center point, opening angle α, and orientation θ

Table 3.1: Exemplary state definitions that apply to the concepts presented in this thesis. Note that all stated position coordinates are within the UTM coordinate system.

3.4 Graph-based optimization

In this section, we explain the essential equations behind graph-based optimization approaches and introduce the notation used throughout the rest of this thesis.

3.4.1 Probability maximization and error minimization

In the following, we describe the relation between the maximum a posteriori (MAP) approach to optimization and the minimization of weighted quadratic errors. We describe the problem of estimating a state vector $\boldsymbol{x}$ that best fits a set of measurements $\boldsymbol{z}$ as the commonly used optimization problem

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p(\boldsymbol{x} \mid \boldsymbol{z}), \tag{3.1}$$

with posterior distribution $p(\boldsymbol{x} \mid \boldsymbol{z})$. In the following, we show how to convert this general formulation into an equivalent minimization problem of weighted quadratic errors. For clarity, we omit the presence of outliers and discuss them separately in Section 3.4.4. Using Bayes rule, we transform the problem into

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p(\boldsymbol{z} \mid \boldsymbol{x}) p(\boldsymbol{x}), \tag{3.2}$$

where $p(\boldsymbol{z} \mid \boldsymbol{x})$ is called the likelihood and $p(\boldsymbol{x})$ the prior distribution. We make the following assumptions about the involved random variables and distributions:

Assumption 1. *Measurements and prior terms are independent and identically distributed (i.i.d.).*

It follows that

$$p(\boldsymbol{x}) = \prod_j p(\boldsymbol{x}_j)$$
$$p(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_i p(\boldsymbol{z}_i \mid \boldsymbol{x}).$$

Assumption 2. *The observation likelihood functions are Gaussian.*

In detail, we assume the distributions to be

$$\begin{aligned} p(\boldsymbol{z}_i \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}_i \mid h_i(\boldsymbol{x}), \Sigma_i\right) &\propto \exp\left(-\frac{1}{2}(\boldsymbol{z}_i - h_i(\boldsymbol{x}))^\top \Sigma_i^{-1}(\boldsymbol{z}_i - h_i(\boldsymbol{x}))\right) \\ &= \exp\left(-\frac{1}{2}\ \mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x}))^\top \Omega_i\ \mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x}))\right), \\ \text{with} \quad \mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x})) &= \boldsymbol{z}_i - h_i(\boldsymbol{x}) \end{aligned}$$

and measurement function $h_i(\boldsymbol{x})$ that produces an expected measurement given the state $\boldsymbol{x}$. The term $\mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x}))$ is commonly called the error function.

Assumption 3. *The prior distributions are either Gaussian or uniform.*

We define the prior distributions either as Gaussians

$$\begin{aligned} p(\boldsymbol{x}_j) = \mathcal{N}\left(\boldsymbol{x}_j \mid \mu_j, \Sigma_j\right) \ &\propto \exp\left(-\frac{1}{2}(\boldsymbol{x}_j - \mu_j)^\top \Sigma_j^{-1}(\boldsymbol{x}_j - \mu_j)\right) \\ &= \exp\left(-\frac{1}{2}\ \mathrm{e}(\boldsymbol{x}_j)^\top \Omega_j \mathrm{e}(\boldsymbol{x}_j)\right) \end{aligned}$$

or as a uniform distribution over all possible states

$$p(\boldsymbol{x}_j) = \epsilon,$$

where we used $\mathrm{e}(\boldsymbol{x}_j)$ to match the notation of the likelihood function. From a practical point of view, the option to choose between a uniform and Gaussian prior for each involved state allows us to apply prior knowledge about states. In our case, we consider the position of a map landmark as a prior, which we describe in more detail in Section 4.3.2. For the following derivation, we assume that the priors are Gaussian. In the case of uniform distributions, the related terms cancel out such that the given derivation stays valid. Using the above assumptions, we

transform Equation (3.2) into

$$\begin{aligned}
\boldsymbol{x}^* &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \prod_i p(\boldsymbol{z}_i \mid \boldsymbol{x}) \prod_j p(\boldsymbol{x}_j) \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} -\log \left(\prod_i p(\boldsymbol{z}_i \mid \boldsymbol{x}) \prod_j p(\boldsymbol{x}_j) \right) \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} -\left(\sum_i \log p(\boldsymbol{z}_i \mid \boldsymbol{x}) + \sum_j \log p(\boldsymbol{x}_j) \right) \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_i \mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x}))^\top \Omega_i \ \mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x})) + \sum_j \mathrm{e}(\boldsymbol{x}_j)^\top \Omega_j \mathrm{e}(\boldsymbol{x}_j).
\end{aligned}$$

For notational convenience, we gather the indices $k = \{i, j\}$ and use the error vector $\boldsymbol{e}_k$ to represent the results of either $\mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x}))$ or $\mathrm{e}(\boldsymbol{x}_j)$. We can now represent the MAP approach from Equation (3.1) as

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_k \boldsymbol{e}_k^\top \Omega_k \boldsymbol{e}_k, \tag{3.3}$$

which shows that under the assumptions A-1, A-2, and A-3 the MAP formulation is equivalent to minimizing a sum over weighted quadratic errors.

3.4.2 Gauss-Newton algorithm

A classical way of solving non-linear least squares problems is to use the Gauss-Newton algorithm. The general idea of Gauss-Newton is to approximate the cost function $\mathrm{F}(\boldsymbol{x})$ locally with an analytically solvable function and then iteratively improve the estimates $\boldsymbol{x}$. In the following, we use the work by Grisetti et al. (2010) and derive the algorithm with respect to our graph-based optimization problem and notation. For notational convenience, we first rewrite our optimization problem as

$$\begin{aligned}
\boldsymbol{x}^* &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_k \boldsymbol{e}_k^\top \Omega_k \boldsymbol{e}_k \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_k \mathrm{e}_k(\boldsymbol{x})^\top \Omega_k \mathrm{e}_k(\boldsymbol{x}) \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} \, \mathrm{F}(\boldsymbol{x}),
\end{aligned}$$

where we use the notation $\mathrm{e}_k(\boldsymbol{x})$ to make the error functions dependent on all state variables and gather all costs in $\mathrm{F}(\boldsymbol{x})$. We use Taylor approximation to locally approximate the cost function $\mathrm{F}(\boldsymbol{x})$ around the linearization point $\breve{\boldsymbol{x}}$ as

$$\begin{aligned}
\mathrm{F}(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}) &= \sum_k \mathrm{e}_k(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x})^\top \Omega_k \mathrm{e}_k(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}) \\
&\approx \sum_k \left(\mathrm{e}_k(\breve{\boldsymbol{x}}) + \mathrm{J}_k(\breve{\boldsymbol{x}}) \Delta\boldsymbol{x} \right)^\top \Omega_k \left(\mathrm{e}_k(\breve{\boldsymbol{x}}) + \mathrm{J}_k(\breve{\boldsymbol{x}}) \Delta\boldsymbol{x} \right),
\end{aligned}$$

with $\mathrm{J}_k(\breve{\boldsymbol{x}})$ being the Jacobian of error function $\mathrm{e}_k(\breve{\boldsymbol{x}})$ at linearization point $\breve{\boldsymbol{x}}$ and the update $\Delta\boldsymbol{x}$. Multiplying out and reordering the terms results in

$$
\begin{aligned}
&\mathrm{F}\left(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}\right) \\
&\approx \sum_k \mathrm{e}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{e}_k(\breve{\boldsymbol{x}}) + 2\mathrm{e}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{J}_k(\breve{\boldsymbol{x}}) \Delta\boldsymbol{x} + \Delta\boldsymbol{x}^\top \mathrm{J}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{J}_k(\breve{\boldsymbol{x}}) \Delta\boldsymbol{x} \\
&= \underbrace{\sum_k \mathrm{e}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{e}_k(\breve{\boldsymbol{x}})}_{c} + 2 \underbrace{\sum_k \mathrm{e}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{J}_k(\breve{\boldsymbol{x}})}_{\mathrm{b}(\breve{\boldsymbol{x}})=\boldsymbol{b}} \Delta\boldsymbol{x} + \Delta\boldsymbol{x}^\top \underbrace{\sum_k \mathrm{J}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{J}_k(\breve{\boldsymbol{x}})}_{\mathrm{H}(\breve{\boldsymbol{x}})=H} \Delta\boldsymbol{x} \\
&= \Delta\boldsymbol{x}^\top H \Delta\boldsymbol{x} + 2\boldsymbol{b}^\top \Delta\boldsymbol{x} + c,
\end{aligned} \tag{3.4}
$$

where we have shown that after Taylor approximation our target function is now quadratic in $\Delta\boldsymbol{x}$. We can now solve for the optimal update $\Delta\boldsymbol{x}^*$ based on the current linearization point $\breve{\boldsymbol{x}}$ in closed form as

$$
\begin{aligned}
\Delta\boldsymbol{x}^* &= \underset{\Delta\boldsymbol{x}}{\operatorname{argmin}} \mathrm{F}\left(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}\right), \\
\frac{\mathrm{F}\left(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}\right)}{\partial \Delta\boldsymbol{x}} &= 2H\Delta\boldsymbol{x} + 2\boldsymbol{b} \overset{!}{=} 0,
\end{aligned}
$$

which leads to the system of linear equations of the form

$$
H\Delta\boldsymbol{x}^* = -\boldsymbol{b}. \tag{3.5}
$$

Usually, this form is not directly solved for $\Delta\boldsymbol{x}^*$ because it requires inverting the system matrix H, which is usually a computationally expensive task. Commonly used methods are, for example, QR-factorization and Cholesky decomposition. It is important to note that the system matrix H and gradient vector $\boldsymbol{b}$ are computed in Equation (3.4) as

$$
H = \sum_k H_k = \sum_k \mathrm{J}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{J}_k(\breve{\boldsymbol{x}}), \tag{3.6}
$$

$$
\boldsymbol{b} = \sum_k \boldsymbol{b}_k = \sum_k \mathrm{e}_k(\breve{\boldsymbol{x}})^\top \Omega_k \mathrm{J}_k(\breve{\boldsymbol{x}}), \tag{3.7}
$$

which means that the individual summands H_k and $\boldsymbol{b}_k$ can be computed independently for each error function. This allows for only partially relinearizing the system and more efficient calculations by grouping error functions based on their mathematical properties. In Section 4.3.2, we make use of this to simplify the terms for map landmark priors. To explicitly point out that the system matrix H and the gradient vector $\boldsymbol{b}$ are computed based on the linearization point $\breve{\boldsymbol{x}}$, we later use the definition

$$
\begin{aligned}
H &= \mathrm{H}\left(\breve{\boldsymbol{x}}\right), \\
\boldsymbol{b} &= \mathrm{b}\left(\breve{\boldsymbol{x}}\right).
\end{aligned}
$$

Another important aspect is that the inverse of the system matrix, i.e., H^{-1}, is the covariance matrix of the Normal distribution

$$\mathcal{N}\left(\breve{\boldsymbol{x}}, H^{-1}\right),$$

which is based on the observation that H is constructed in Equation (3.6) by propagating the uncertainty of measurement errors into the state space of $\breve{\boldsymbol{x}}$ (Grisetti et al., 2010). After computing the optimal state update $\Delta\boldsymbol{x}^*$, we apply it to the linearization point $\breve{\boldsymbol{x}}$, which completes one Gauss-Newton iteration and results in the optimized state vector

$$\boldsymbol{x}^* = \breve{\boldsymbol{x}} + \Delta\boldsymbol{x}^*. \tag{3.8}$$

In summary, Algorithm 1 describes the different required steps.

1 set $\breve{\boldsymbol{x}}$ to initial estimate;
2 **while** *not converged* **do**
3 compute $H = \mathrm{H}(\breve{\boldsymbol{x}})$ and $\boldsymbol{b} = \mathrm{b}(\breve{\boldsymbol{x}})$;
4 solve $H\Delta\boldsymbol{x}^* = -\boldsymbol{b}$;
5 apply update $\boldsymbol{x}^* = \breve{\boldsymbol{x}} + \Delta\boldsymbol{x}^*$;
6 set $\breve{\boldsymbol{x}} = \boldsymbol{x}^*$;

Algorithm 1: Gauss-Newton optimization

Levenberg-Marquardt algorithm

A popular alternative to Gauss-Newton is the Levenberg–Marquardt (LM) algorithm. It is based on dampening the system matrix H in Equation (3.5), which is rewritten as

$$(H + \lambda\mathbf{I})\,\Delta\boldsymbol{x}^* = -\boldsymbol{b},$$

with the dampening parameter λ. The intuition behind the dampening parameter is that the higher its value, the smaller is the resulting update $\Delta\boldsymbol{x}^*$. The value of the dampening parameter is typically chosen based on some heuristic. A simple one is to adjust it in each iteration after computing the parameter update $\Delta\boldsymbol{x}^*$. If an update does not improve the overall cost, i.e., $\mathrm{F}(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}^*) > \mathrm{F}(\breve{\boldsymbol{x}})$, the computed update can be considered as too greedy. In this case, the computed update is ignored and the whole iteration redone with an increased dampening value λ. In the case that the update successfully reduces costs, i.e., $\mathrm{F}(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}^*) < \mathrm{F}(\breve{\boldsymbol{x}})$, the dampening value λ can be decreased to improve the convergence speed.

Compared to Gauss-Newton, LM is slower as it might require more iterations to converge. Also, due to rejecting suboptimal updates, LM is guaranteed to converge. This is not the case for Gauss-Newton, which might even diverge such that optimization fails. A benefit of LM is that it can also be used for applications in which the equation system is underconstrained. Nevertheless, we still prefer Gauss-Newton for research purposes and development because it more easily reveals implementation and conceptual flaws as it does not hide rank-deficiency issues.

3.4.3 Gauss-Newton on manifolds

A practical issue of the Gauss-Newton state update $\boldsymbol{x}^* = \breve{\boldsymbol{x}} + \Delta\boldsymbol{x}^*$ (Equation 3.8) is that it assumes Euclidean spaces for the involved states. This renders other state space definitions without further adjustments incompatible. A relevant example for a state space that is not Euclidean is the vehicle orientation, which is defined as $\theta \in [-\pi, \pi)$. Using the state update equation might therefore produce results with the vehicle orientation being outside its defined limits. A tailored solution to our simple example is adding or subtracting 2π from θ whenever it exceeds the limit, which is called normalization. The more generalized notation for handling state space mapping issues is to use manifold operations. Rewriting Equation (3.8) with the manifold operation $\boxplus$ yields

$$\boldsymbol{x}^* = \breve{\boldsymbol{x}} \boxplus \Delta\boldsymbol{x}^*.$$

The manifold operator $\boxplus$ can be interpreted as a mechanism that ensures that the update $\Delta\boldsymbol{x}^*$ is correctly applied, and $\boldsymbol{x}^*$ is still in a valid state space.

Besides applying state updates, manifolds are also useful within error functions. We demonstrate this with the error function $\mathrm{e}(z, \theta) = z - \theta$, where $z \in [-\pi, \pi)$ is the measurement of the vehicles orientation and θ is the current expected measurement. This error function might produce high errors if z and θ are on the opposite ends of the state limits, although both are close together. This is again solved by using manifold operations within the error function instead of relying on subtraction. Figure 3.5 illustrates this issue.

We here demonstrated manifold operations for the orientation of a SE(2) vehicle pose as this relates to our use case. Manifolds are also especially required for SE(3) poses, in which the angular state components are additionally prone to singularities. For a more in-depth discussion of the mathematical background, we refer to the work of Hertzberg (2008) and Hertzberg et al. (2013).

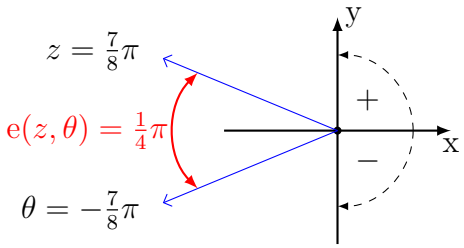


Figure 3.5: The illustration shows the requirement for using manifolds within angular state spaces. The angular measurement z and the expected measurement θ are both at the end of the state space limits $[-\pi, \pi)$. Simple subtraction within the error function produces the incorrect error $\mathrm{e}(z, \theta) = z - \theta = \frac{7}{4}\pi$. Using manifold operations ensures that the error is correctly mapped across the space limits and yields $\mathrm{e}(z, \theta) = z \boxminus \theta = \frac{1}{4}\pi$.

3.4.4 Handling outliers

An essential aspect of robust optimization is handling outliers, which might have a drastic effect on the optimization result if being unhandled. In our case, outliers are measurements that violate our Gaussian assumptions A-2 and A-3. One way of dealing with outliers is carefully selecting the constraints included in the optimization with the intention of filtering out outliers beforehand. We discuss this option for our problem in Section 4.1. Another approach is to alleviate the effects of outliers within the optimization by weighting down large error terms. This is referred to as M-Estimation (Huber and Ronchetti, 2009). The original optimization problem from Equation (3.3) is modified as

$$\begin{aligned} \boldsymbol{x}^* &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_k \rho\left(\boldsymbol{e}_k(\boldsymbol{x})^\top \Omega_k \boldsymbol{e}_k(\boldsymbol{x})\right), \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_k \rho\left(F_k(\boldsymbol{x})\right), \end{aligned} \tag{3.9}$$

where ρ is called a robust cost function that is positive-definite and symmetric (Zhang, 1997). Depending on the characteristics of the involved cost terms, different robust cost functions are convenient. To illustrate the concept, we briefly state dynamic covariance scaling (DCS) by Agarwal et al. (2013) based on the formulas by MacTavish and Barfoot (2015), who write the robust cost function as

$$\rho_{\mathrm{DCS}}\left(F_k(\boldsymbol{x})\right) = \begin{cases} F_k^2/2 & F_k^2 \leq \phi \\ \frac{2\phi F_k^2}{F_k^2 + \phi} - \frac{\phi}{2} & F_k^2 > \phi, \end{cases}$$

where we use F_k as an abbreviation for $F_k(\boldsymbol{x})$ and ϕ is a tuning parameter that controls the impact of the robust cost function. A more classical robust cost

function is the Cauchy kernel (also called Lorentzian), defined as

$$\rho_{\text{Cauchy}}\left(F_k(\boldsymbol{x})\right) = \frac{\phi^2}{2}\log\left(1 + \frac{F_k^2}{\phi^2}\right).$$

We refer to Zhang (1997) for a more comprehensive list of further robust cost functions. Following Zhang (1997) and Agarwal (2015), we interpret Equation (3.9) as an iterative reweighted least squares (IRLS) problem and rewrite it as

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_k \boldsymbol{e}_k(\boldsymbol{x})^\top \left(w\left(F_k(\breve{\boldsymbol{x}})\right)\Omega_k\right)\boldsymbol{e}_k(\boldsymbol{x}),$$

where w is a weighting function, and $\breve{\boldsymbol{x}}$ is the linearization point and result of the last Gauss-Newton iteration. We refer to Förstner and Wrobel (2016, Chapter 4.7) for a more in-depth discussion on weighted least squares for robust estimation. For DCS, the weighting function is given as

$$w_{\text{DCS}}\left(F_k(\breve{\boldsymbol{x}})\right) = \begin{cases} 1 & F_k^2 \leq \phi \\ \frac{4\phi^2}{(F_k^2+\phi)^2} & F_k^2 > \phi \end{cases},$$

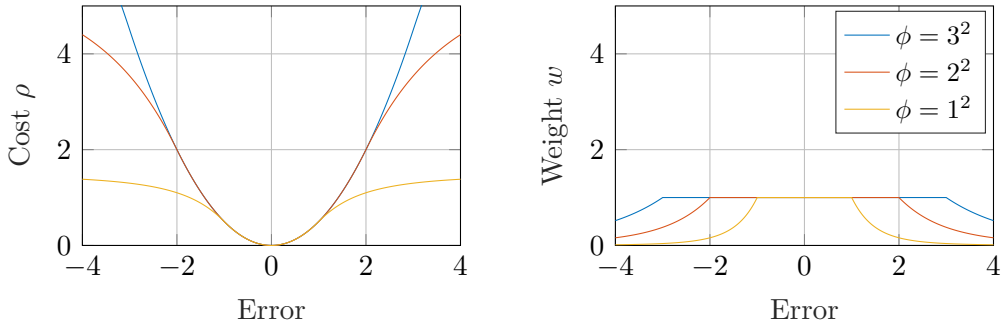
with tuning parameter ϕ, which is the quadratic error threshold up to which DCS is similar to standard least squares. This means that we can control the error boundary b by setting

$$\phi = b^2,$$

which allows an intuitive interpretation of the DCS effects. The weighting function for the Cauchy robust cost function is

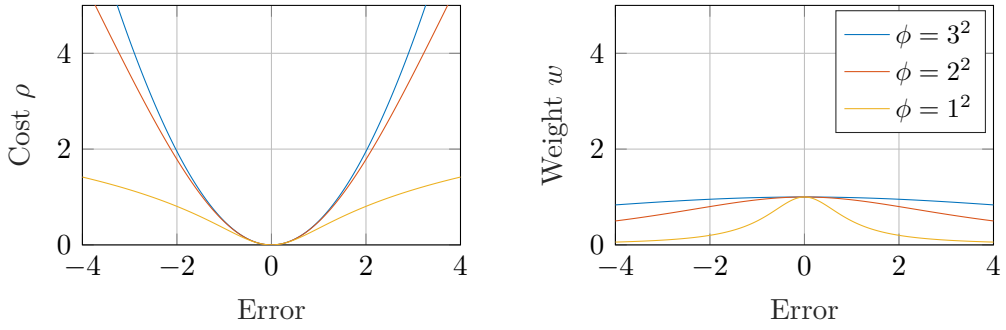
$$w_{\text{Cauchy}}\left(F_k(\breve{\boldsymbol{x}})\right) = \frac{1}{1 + \frac{F_k^2}{\phi^2}}.$$

We illustrate the robust cost and the weighting function for the DCS and Cauchy kernel in Figure 3.6. Choosing a suitable cost function depends on the nature of the data and the underlying optimization problem. DCS is popularly used for robustifying data association constraints. The underlying weighting effectively deactivates outliers and is therefore commonly used to robustify loop closure decisions in SLAM applications. As we will discuss in Chapter 4.1, our localization architecture carefully selects the integrated data association constraints, such that using DCS is not beneficial in our case. In contrast to other approaches using DCS might, in our case, even prevent our graph-based approach from converging to the correct solution. Therefore, we make use of the Cauchy function for all included constraints in the graph. We consider this a trade-off between outlier rejection and allowing the optimization to jump to the correct solution. This is the case in, for example, reinitialization phases.



(a) DCS cost function

(b) DCS weight function



(c) Cauchy cost function

(d) Cauchy weight function

Figure 3.6: Illustration of DCS and Cauchy robust cost as well as their corresponding weight functions. The parameter ϕ allows controlling the cost function's behavior. In the DCS case, it allows to set the boundary up to which the errors are equal to standard least squares.

3.5 Factor graph representation

Regardless of choosing either the MAP or error minimization interpretation for SLAM problems (see Section 3.4.1), it can be helpful to represent the optimization problem graphically. Figure 3.7 shows two commonly used representations. In both representations, the unfilled nodes represent unknown states that we want to estimate. The difference between both representations is how they outline relations between the states and measurements. In the dynamic Bayesian network, nodes are connected through arrows, representing causal dependencies, and grayed out states are used to denote known measurements. On the other side, the factor graph representation outlines the error relation between states. For a more detailed discussion of the various aspects of graphical representations, we refer to the work of Dellaert and Kaess (2017) and Förstner (2013). In the scope of this thesis, we use factor graphs and the following factor definition.

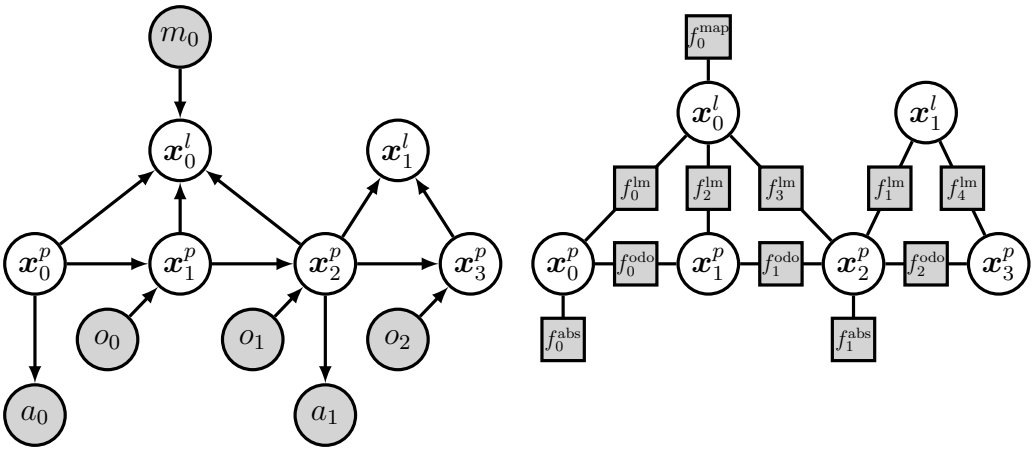
Definition 2. *A factor f_i^k is a triplet composed of*

- *a measurement $\boldsymbol{z}_i^k$,*
- *the corresponding covariance matrix Σ_i^k or information matrix $\Omega_i^k = \Sigma_i^{k-1}$,*

and

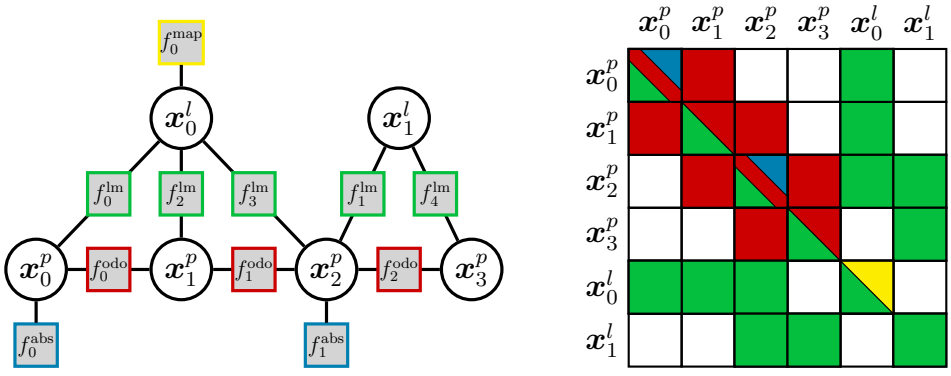
- *an error function* $\mathrm{e}_k(\boldsymbol{x}, \boldsymbol{z}_i^k)$ *between the state and the measurement vector.*

Within the definition, the superscript k denotes the type of the factor, for example, an absolute pose measurement from GNSS or relative measurement of a pole landmark. The subscript i is the index within all available measurements of the corresponding type. Within this thesis, we use the factor graph representation because its notation directly corresponds to the structure of the system matrix H. Figure 3.8 gives an example.



(a) Dynamic Bayesian network (b) Factor graph

Figure 3.7: Comparison of two commonly used graph representations. Both representations describe the same state estimation problem with vehicle poses $\boldsymbol{x}^p = \{\boldsymbol{x}_0^p, ..., \boldsymbol{x}_3^p\}$ and two landmarks $\boldsymbol{x}^l = \{\boldsymbol{x}_0^l, \boldsymbol{x}_1^l\}$, where only $\boldsymbol{x}_0^l$ is in the map.



(a) Factor graph (b) System matrix H

Figure 3.8: Factor graph and its corresponding system matrix structure. (a) A factor graph with landmark factors in green, odometry factors in red, absolute factors in blue, and a yellow map factor. (b) Structure of the corresponding system matrix H within the optimization problem. The coloring shows the contributions of the different factor types to the matrix.

3.6 Marginalization

A common way of keeping an optimization problem computationally tractable is to limit the state vector's size. This can be achieved by either simply deleting states or by marginalizing them out. While we discuss both options in Section 5.3.2, we state the mathematical background for marginalization in the following. The overall marginalization problem is given as

$$p(\boldsymbol{x}_n) = \int p(\boldsymbol{x}_n, \boldsymbol{x}_m) d\boldsymbol{x}_m, \tag{3.10}$$

where $p(\boldsymbol{x}_n, \boldsymbol{x}_m) = p(\boldsymbol{x})$ is the distribution over all states, $\boldsymbol{x}_m$ contains the marginalized states, and $p(\boldsymbol{x}_n)$ is the marginalized distribution without the states $\boldsymbol{x}_m$. If the joint distribution $p(\boldsymbol{x}_n, \boldsymbol{x}_m)$ is Gaussian and given in covariance form as

$$p(\boldsymbol{x}_n, \boldsymbol{x}_m) \sim \mathcal{N}\left(\begin{bmatrix} \mu_m \\ \mu_n \end{bmatrix}, \begin{bmatrix} \Sigma_m & \Sigma_{mn} \\ \Sigma_{mn}^\top & \Sigma_n \end{bmatrix} \right),$$

marginalization of $\boldsymbol{x}_m$ is equal to deleting all entries involving m and results in

$$p(\boldsymbol{x}_n) \sim \mathcal{N}\left(\mu_n, \Sigma_n\right),$$

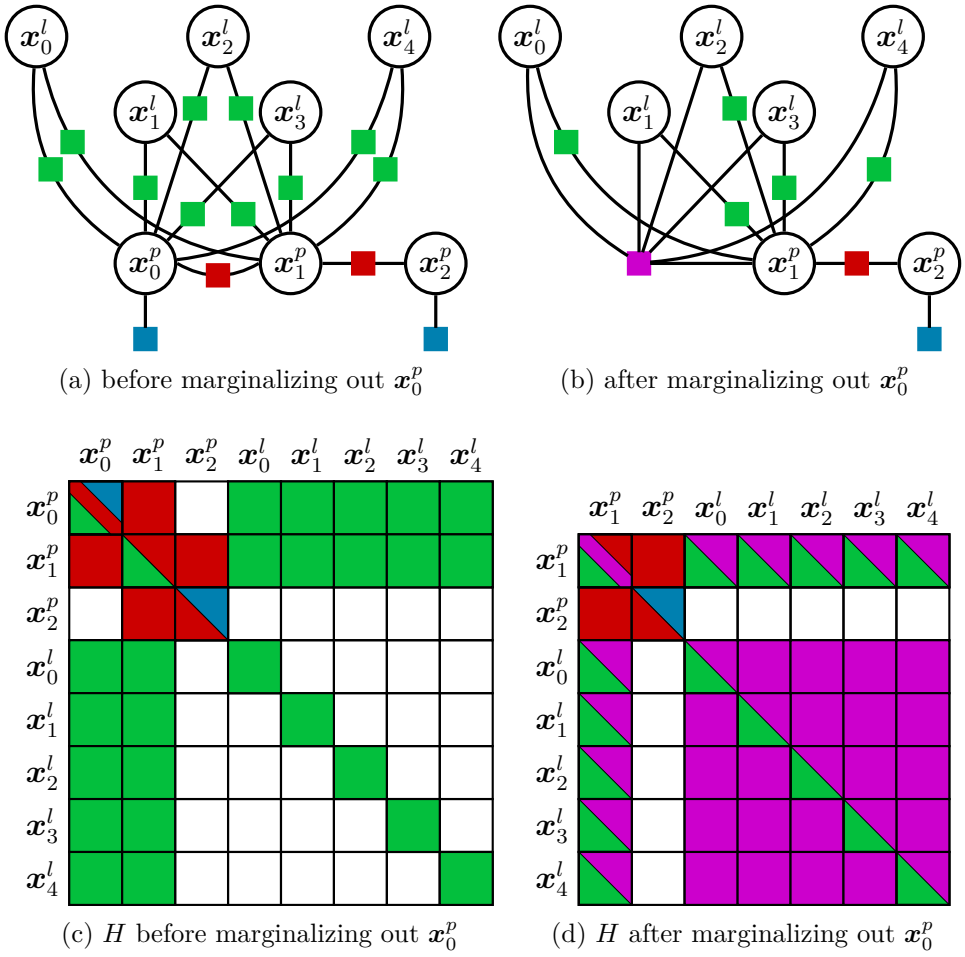
which is explained by Walter et al. (2007) in more detail. Since it is beneficial in graph-based SLAM to represent Gaussians with mean and information matrix instead of mean and covariance matrix, marginalization must be handled differently. Assuming that the joint distribution from Equation (3.10) is denoted as

$$p(\boldsymbol{x}_n, \boldsymbol{x}_m) \sim \mathcal{N}\left(\begin{bmatrix} \mu_m \\ \mu_n \end{bmatrix}, \begin{bmatrix} \Omega_m & \Omega_{mn} \\ \Omega_{mn}^\top & \Omega_n \end{bmatrix}^{-1} \right),$$

the Schur complement can be used to calculate the marginal distribution

$$\begin{aligned} p(\boldsymbol{x}_n) &\sim \mathcal{N}\left(\mu_n, \Omega_n^{-1}\right), \\ \Omega_n^{-1} &= \left[\Omega_n - \Omega_{mn}\Omega_m^{-1}\Omega_{mn}^\top\right]^{-1}. \end{aligned} \tag{3.11}$$

A major drawback of this exact marginalization is that the sparsity pattern of the resulting information matrix Ω_n is negatively influenced and thus leads to an increased computational demand for most subsequent operations involving Ω_n. We refer to Sibley et al. (2010), who explain in more detail how the sparsity pattern changes. To illustrate the change in sparsity, Figure 3.9 compares a graph before and after marginalization.



(a) before marginalizing out $\boldsymbol{x}_0^p$ (b) after marginalizing out $\boldsymbol{x}_0^p$

(c) H before marginalizing out $\boldsymbol{x}_0^p$ (d) H after marginalizing out $\boldsymbol{x}_0^p$

Figure 3.9: The figure illustrates a worst-case example of marginalization. It shows an exemplary factor graph and its corresponding system matrix before and after marginalization. (a) The original factor graph before marginalization. (b) All states that were connected to $\boldsymbol{x}_0^p$ are now interconnected through a dense marginalization prior. (c) The system matrix H has several empty entries, which are colored in white. (d) After marginalization, the system matrix H is significantly more dense than before. All entries that are affected by the marginalization are colored in purple. Note that the entries for vehicle state $\boldsymbol{x}_2^p$ are not influenced.

3.6.1 Relation to iterative optimization

To show the connection between marginalization using Schur complement and iterative optimization, we rewrite the equation system that needs to be solved in each Gauss-Newton iteration (Equation 3.5) as

$$\begin{bmatrix} H_a & H_{ab} \\ H_{ab}^\top & H_b \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}_a^* \\ \Delta \boldsymbol{x}_b^* \end{bmatrix} = - \begin{bmatrix} \boldsymbol{b}_a \\ \boldsymbol{b}_b \end{bmatrix}, \tag{3.12}$$

where we have split up the involved matrices and vectors into the two parts a and b. Part a corresponds to the part of the state vector that we want to marginalize out, whereas part b is the one that we want to keep. It is important to point out that the system matrix $H = \mathrm{H}(\breve{\boldsymbol{x}})$ and gradient vector $\boldsymbol{b} = \mathrm{b}(\breve{\boldsymbol{x}})$ are constructed based on the linearization point $\breve{\boldsymbol{x}}$. We will later use this to elaborate on the linearization errors induced by marginalization. Another essential aspect to remember is that H^{-1} is the covariance matrix of the Normal distribution $\mathcal{N}(\breve{\boldsymbol{x}}, H^{-1})$, which shows the relation to the probabilistic interpretation from the previous Section 3.6. Following Sibley et al. (2010), we now rearrange Equation (3.12) into

$$\begin{bmatrix} H_a & H_b \\ 0 & H_b - H_{ab}^\top H_a^{-1} H_{ab} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}_a^* \\ \Delta \boldsymbol{x}_b^* \end{bmatrix} = - \begin{bmatrix} \boldsymbol{b}_a \\ \boldsymbol{b}_b - H_{ab}^\top H_a^{-1} \boldsymbol{b}_a \end{bmatrix},$$

such that we can partly solve the equation system for $\Delta \boldsymbol{x}_b^*$ without solving for $\Delta \boldsymbol{x}_a^*$ as

$$\left(H_b - H_{ab}^\top H_a^{-1} H_{ab}\right) \Delta \boldsymbol{x}_b^* = - \left(\boldsymbol{b}_b - H_{ab}^\top H_a^{-1} \boldsymbol{b}_a\right).$$

Comparing the left-hand side of the equation system with Equation (3.11) reveals that we have applied the Schur complement, which is, as previously shown, equivalent to marginalization. Gathering all parts of the equation that contain information about the marginalized states yields

$$\left(H_b - H_{\text{marg}}\right) \Delta \boldsymbol{x}_b^* = - \left(\boldsymbol{b}_b - \boldsymbol{b}_{\text{marg}}\right).$$

We rewrite the equation and explicitly denote the dependency on the linearization point as

$$\left(\mathrm{H}_b(\breve{\boldsymbol{x}}_b) - \mathrm{H}_{\text{marg}}\left(\begin{bmatrix} \breve{\boldsymbol{x}}_a \\ \breve{\boldsymbol{x}}_b \end{bmatrix}\right)\right) \Delta \boldsymbol{x}_b^* = - \left(\mathrm{b}_b(\breve{\boldsymbol{x}}) - \mathrm{b}_{\text{marg}}\left(\begin{bmatrix} \breve{\boldsymbol{x}}_a \\ \breve{\boldsymbol{x}}_b \end{bmatrix}\right)\right),$$

where we can see that H_{marg} and $\boldsymbol{b}_{\text{marg}}$ depend on the linearization point $\breve{\boldsymbol{x}}$ of the current Gauss-Newton iteration. Linearization errors now emerge in subsequent iterations of the Gauss-Newton algorithm because H_{marg} and $\boldsymbol{b}_{\text{marg}}$ are not recomputed but treated as constant terms. In return, this ensures that the state update

no longer requires linearizing $\boldsymbol{x}_a$ as it is not part of the optimization anymore. In comparison, all states within $\boldsymbol{x}_b$ remain active such that the corresponding system matrix H_b and gradient vector $\boldsymbol{b}_b$ can not only be relinearized but also be augmented with additional information that is available to the system at a later point in time. A major drawback besides the induced linearization errors is that the fill-in produced by H_{marg} renders the overall system matrix H dense and therefore negatively impacts the required computation time for solving the linear equation system. In Chapter 5.3.2, we propose a method for sliding window optimization that limits linearization errors and simultaneously prevents fill-in.

Chapter 4

Localization on general-purpose landmark maps

In the following, we present our approach to vehicle localization for automated driving. We extend upon our previously reported findings (Wilbers et al., 2019a) and discuss the aspects of our approach in greater detail. In this chapter, our main contribution is the adaptation of graph-based optimization such that it is suitable for self-localization in the context of automated driving. In detail, the contributions that we present in this chapter include an architecture for graph-based sliding window localization, a data association approach that utilizes general-purpose third-party landmark maps, and a comparison to particle filters for pose estimation. Our key claims are that our graph-based localization approach

(i) provides globally accurate pose estimates,

(ii) incorporates landmark measurements in a generic sensor-independent fashion,

(iii) utilizes general-purpose third-party landmark maps,

(iv) integrates delayed measurements,

(v) revises map associations to increase the localization quality, and

(vi) is fast and frequent enough for application in an automated vehicle.

We focus on the localization aspects of our approach in the following and cover map refinements separately in Chapter 5. Our graph-based localization approach relies on a fixed pose frequency sliding window graph for computational tractability, for which we first present our design principles in Section 4.1 and afterward present our approach in Section 4.2. Furthermore, we propose an approach for integrating third-party maps in our sliding window graph in Section 4.3 and present

our data association technique in Section 4.4. Our data association approach includes locally associating measurements to find out if they belong to the same landmark, matching landmarks to a given general-purpose map, and temporally filtering map associations. We discuss each of these steps in detail. Furthermore, we derive and discuss the error functions required for different types of landmarks in Section 4.5 and discuss the different options for synchronizing measurement data in Section 4.6. We conclude this chapter by comparing our sliding window approach to particle filters on an argumentative level in Section 4.7.

4.1 Design principles

Our localization approach fuses data from GNSS, odometry, landmark detectors, and general-purpose landmark maps to estimate a vehicle pose with graph optimization. The architecture of our approach consists of different processing layers that we structure such that they contribute to fulfilling the requirements for our localization system as listed in Section 1.2. In brief, we require highly accurate poses that are computed computationally tractable for online localization with a target frequency between 10 Hz to 20 Hz. Also, our localization system must incorporate landmarks in a generic way such that multiple sensor modalities are integrable. Our approach must incorporate general-purpose map landmarks and fallback to fusing GNSS and odometry in environments without landmarks. Furthermore, our approach must be capable of computing conservative in-vehicle map refinements as update hypotheses that can be transmitted to a back-end service. We target to realize our approach based on design principles that we discuss in the following.

Our approach for designing a robust and reliable localization system is built on abstracting the used sensors away from the core localization system. The idea is that different sensors can detect the same landmarks such that we can incorporate landmark measurements in a general fashion. For example, a lamp post can be measured with a LiDAR, as well as a camera or radar. We suggest using a detection module for each sensor that processes the raw data and outputs landmark measurements in the VRF. The benefits of using generic interfaces are also common in other ADAS domains, e.g., Kubertschak et al. (2014) and Munz et al. (2010). A benefit of using landmarks for localization is that it allows us to divide the raw data processing (i.e., landmark detection) across different processing units in the vehicle such that we can implement a decentralized architecture. This enables us to balance the computational workload and bandwidth requirements of our system. Darms and Winner (2005) discuss this aspect in a more general fashion. To avoid correlated errors and systematic failure, it is crucial that the detectors work individually on the sensor data and do not rely on, e.g., the odo-

metry that is later fused into the state estimate. Additionally, to support the i.i.d. assumption (see Assumption 1 in Chapter 3.4.1), the detectors should be designed without internal tracking or predictions. Whenever we use black box detectors, this is sometimes hard to ensure. In practice, we ignore violations of the i.i.d. assumption for landmark measurements, as the impact heavily depends on the error characteristics of the underlying sensor. We consider it the detector's responsibility to deliver landmark measurements free of systematic and correlated errors. We will show in our evaluation that this works well in practice for our use case of localization in urban areas. It is worth mentioning that advanced methods could be used to handle correlated errors within optimization (see Noack et al., 2015) or even do online calibration to handle systematic errors. While this is common and helpful for regular SLAM systems, we advise against doing so in localization systems for automated driving. Especially online calibration involves the risk of actually decalibrating the sensor parameters in challenging environment scenarios. Related to functional safety, we consider this type of error source too hazardous and therefore exclude online calibration for our approach in this thesis. Overcoming these reservations is an open research field with methods that might help to relax the mentioned issues in the future.

A major design aspect that we consider in our architecture is to avoid any feedback loops, which means that we do not use pose estimates from preceding algorithm cycles as an input to the current or following algorithm cycles. In comparison, it is a common technique in related work to explicitly rely on given pose estimates to, e.g., identify outliers and perform data association, which is prone to error accumulation and may result in failure (Lajoie et al., 2019). As we consider localization for automated driving safety-relevant, it is important that any errors do not amplify over time. In the worst-case, this would lead to a complete failure of the localization system. Therefore, we avoid using estimation results from previous algorithm cycles as a direct input to any of the processing steps in our architecture for subsequent algorithm cycles. To be correct, one exception to this is our map matching, which requires one initial starting pose in order to limit the search area in the map. However, the initial pose could also be a pose from GNSS or any other pose estimate, as it must only be roughly in the area of the true pose. We still consider this feedback-free since the pose must only be approximately correct and errors do not propagate through the system.

A further design principle is that we avoid solving data association within the graph optimization but instead rely on estimating it during graph construction. Although solving data associations, e.g., loop closures and measurement association, is commonly done during graph optimization in SLAM algorithms, we consider it a risk for our localization use case. This is because solving data association within graph optimization might involve the risk of divergence. Instead, we

prefer validating our decisions upfront during graph construction, which enables us to make more comprehensible and conservative decisions.

Another design aspect is that we perform map matching independently of previously found matches. We always search for the best map matching based on the measurements in the current sliding window. This is important because we add and remove measurements in each algorithm cycle from the sliding window such that the map matching results might differ between subsequent algorithm cycles. To avoid matching errors, we additionally filter map associations temporally, which gives us the most probable matches over time. In combination, this allows us to recover whenever the vehicle drove through an area without landmarks, no landmark measurements were available, or the localization was erroneous. We explain our algorithm in detail in Section 4.4.2.

Our architectural choices are based on the discussed design principles. Figure 4.1 illustrates the described elements of our approach and states the references to the related sections in which we provide detailed information. In the following, we briefly describe the illustrated data flow of our architecture. The input data to our approach consists of GNSS, odometry, map data, and a combination of LiDAR, camera, and/or radar. The data from LiDAR, camera, and radar is consumed by given landmark detector modules that provide landmark detections in VRF. Hereby, we completely abstract the landmark detections away from their initial sensor modality such that subsequent modules have no knowledge about the sensor origin. Our local association module accumulates odometry data and landmark measurements to identify which landmark measurements belong to the same landmark. The result of our local association step are clustered landmark measurements that we in a later step include in the graph and a local map based on accumulated odometry data that we use for map matching. In the following map matching step, we calculate the best fit between the local map based on the input data to the current algorithm cycle and the general-purpose map. Subsequently, we accumulate the found map matches over all previous algorithm cycles in our temporal association step to find the best map associations that we include in our graph. In the last three steps of our architecture, we first synchronize all measurements, afterward construct the graph based on the previously found associations, and finally optimize our sliding window graph.

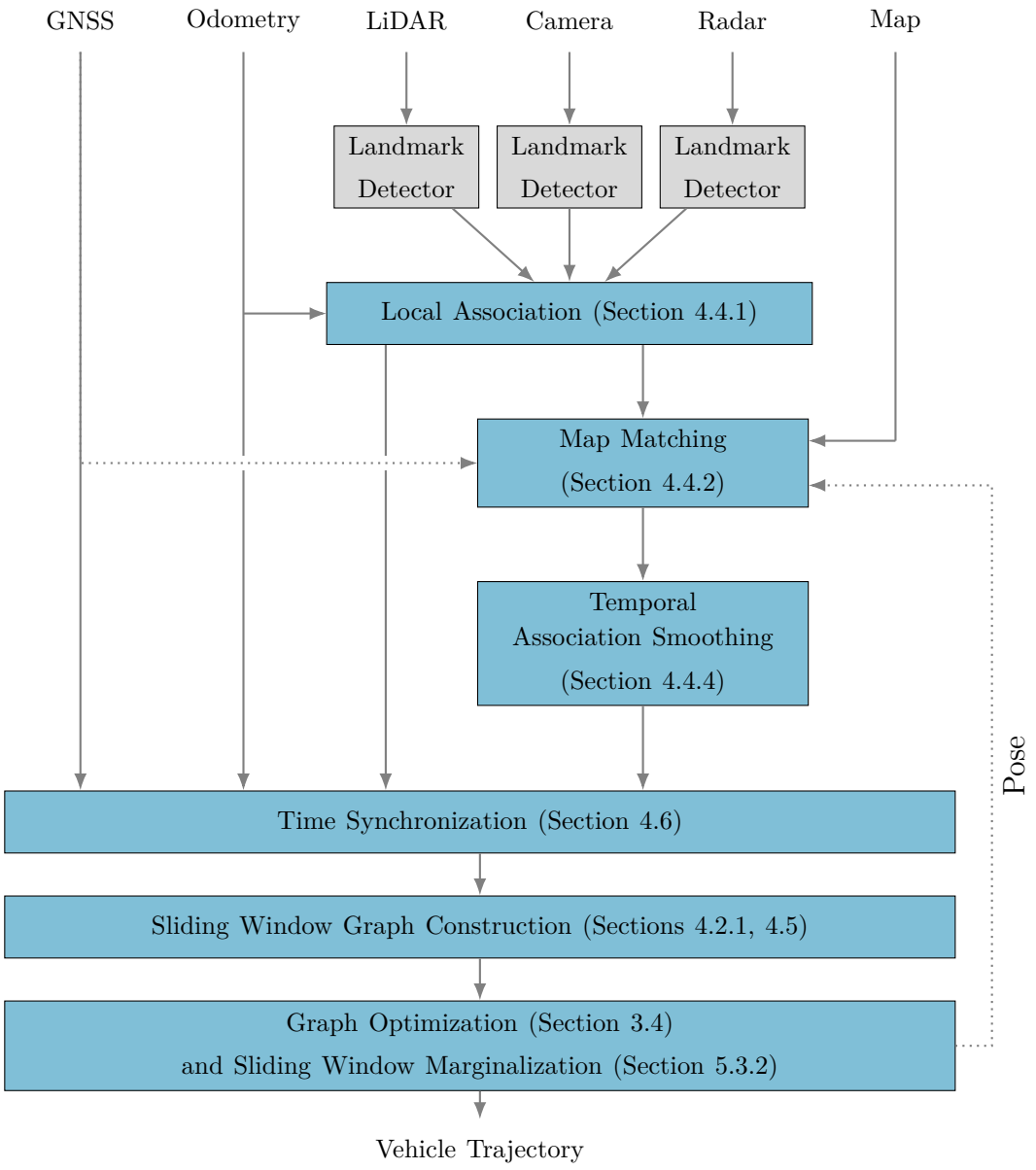


Figure 4.1: The architecture gives an overview of the involved steps in our localization architecture. The highlighted elements in blue are in the scope of this thesis and presented in more detail in the denoted sections.

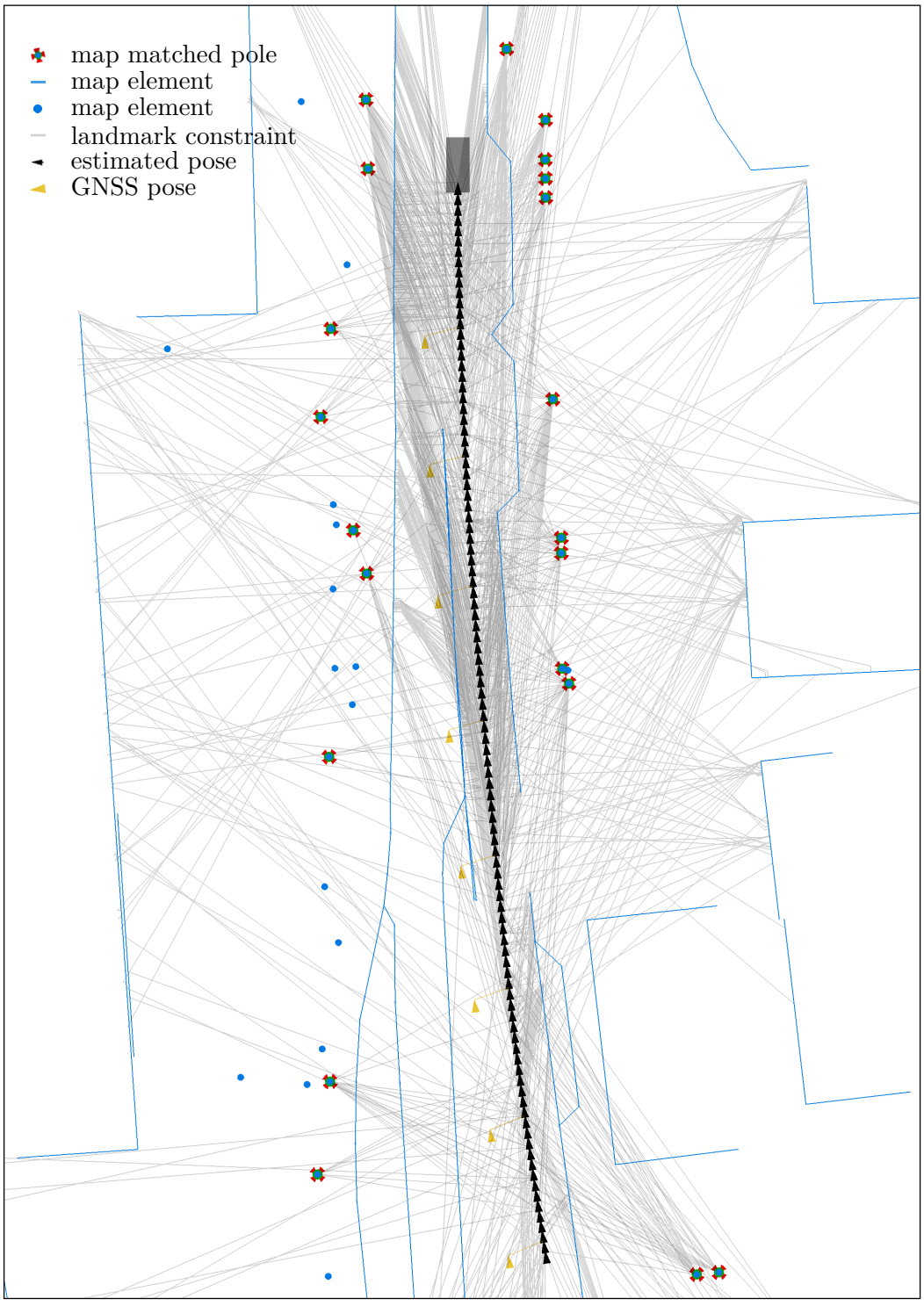


Figure 4.2: The figure shows a screenshot of our sliding window graph implementation in a real-world urban scenario. In this example, we use pole and building facade measurements from LiDAR, lane boundary measurements from camera, odometry measurements, GNSS poses, and map data to constrain the vehicle trajectory. A gray line denotes a constraint between a vehicle pose and a landmark. The number of incorporated landmark constraints in our sliding window graph is naturally limited by the number of landmarks in the environment, the detector's frequencies, and the number of poses in the graph.

4.2 Graph-based sliding window localization

We adopt the graph optimization principle to a sliding window formulation such that it is computationally tractable for online localization. First, we concentrate on the properties, advantages, and disadvantages of our sliding window design. In its most basic form, without including any landmark states and measurements, we derive our sliding window principle from the pose graph formulation of Merfels (2018). We extend upon this by incorporating landmark measurements and including landmarks in the state vector. Although we include landmarks in the state vector, our approach is not a regular SLAM system because of two reasons. The first one is that we never directly change or update the general-purpose map at runtime and only integrate it as prior knowledge. Secondly, we only use the estimated landmark positions as update hypotheses that we can send to a back-end service for validation. Our system forgets about the estimated landmarks as soon as they are out of range and potentially transferred to the back-end service. Therefore, our approach is mainly a localization approach that also allows map refinement instead of a regular SLAM technique. We discuss map refinements separately in Chapter 5. In general, estimating a trajectory is known as smoothing, whereas filtering techniques, like Kalman filter, usually only estimate the latest pose (Sibley et al., 2010). We discuss the benefits of our approach in the remaining sections of this chapter.

4.2.1 Sliding window definition

We define the length of our sliding window based on the number of maximum poses inside the graph. The poses inside the graph have a fixed temporal distance that is equal for all poses. The main advantage of having a sliding window defined over time instead of space is the minimal data maintenance effort. Poses can be stored in a list, and whenever the sliding window is moved, new poses can be added in the front, and old ones can be deleted at the back. More importantly, it also naturally limits the number of landmark measurements that we include in the graph. The total number of landmark measurements depends on the detectors' measurement frequency and the number of visible landmarks at the vehicle location. On the contrary, a sliding window defined by space would require a method that prevents excessive growth of measurements in the graph when, e.g., standing still or driving in circles. We illustrate an exemplary sliding window graph in a real-world urban scenario in Figure 4.2. We denote the total time span that our sliding window graph covers as

$$T = \frac{N}{f},$$

with the frequency of poses f and number of maximum poses N inside the graph. The pose frequency f here describes the temporal resolution of the sliding window graph which can be chosen freely. We denote the state vector for a sliding window at time τ as

$$\boldsymbol{x}_\tau = [\boldsymbol{x}^p_{\tau-T}, \dots, \boldsymbol{x}^p_\tau, \boldsymbol{x}^l_1, \dots, \boldsymbol{x}^l_M]^\top,$$

with poses $\boldsymbol{x}^p$, landmarks $\boldsymbol{x}^l$, and the number of landmarks in the graph M. Depending on how we choose the pose frequency f and the number of maximum poses N, we influence the size of the state vector. At the same time, the state vector size influences the computation time, which must be considered. Additionally, as we want to make sure that subsequent pose vertices are linked, we require that the measurement frequency of the odometry must be higher than the pose frequency f. This guarantees that all subsequent poses are connected through factors in the graph, and no split graph occurs.

It is important to note that the graph's internal resolution, which is defined by the pose frequency f and the number of maximum poses N, is independent of how often and when we optimize the graph. We separately define an optimization frequency f^o that controls how often the algorithm cycle is triggered and the graph optimized. This allows us to distinguish between the requirements within graph construction and the pose output frequency required by subsequent modules. Besides, it is essential to consider that our sliding window's pose frequency directly influences the magnitude of measurement interpolation errors through time synchronization, which we discuss in more detail in Section 4.6. We illustrate the connection between internal graph resolution and algorithm cycle in Figure 4.3.

In addition to the states, we denote the set of raw measurements $\tilde{\boldsymbol{z}}$ as

$$\tilde{\boldsymbol{z}}_\tau = [\tilde{\boldsymbol{z}}_{\tau_1}, \dots, \tilde{\boldsymbol{z}}_{\tau_R}],$$

where R is the number of total measurements within the time span of the sliding window graph τ, and the subscript only iterates the measurements, as multiple measurements for a specific timestamp are allowed. Here $\tilde{\boldsymbol{z}}$ denotes the set of raw measurements with their original timestamp.

To be included in the graph, the measurements must be synchronized to the pose timestamps in $\boldsymbol{x}_\tau$. We denote the synchronized measurements as

$$\boldsymbol{z}_\tau = [\boldsymbol{z}_{\tau_1}, \dots, \boldsymbol{z}_{\tau_G}],$$

where G is the number of measurements in the graph, which is different from the number of raw measurements R as we might exclude landmarks if they are not matched to the map. Additionally, we use the notation $\tilde{\boldsymbol{z}}_{\tau^+}$ and $\boldsymbol{z}_{\tau^+}$ to denote the set of new measurements that we add to the sliding window τ and use $\tilde{\boldsymbol{z}}_{\tau^-}$,

$\boldsymbol{z}_{\tau^-}$ to denote the sets of measurements that we remove from the sliding window graph in algorithm cycle τ.

It is important to note that only the most recent vehicle pose in each sliding window graph is relevant for other modules of the driving stack. Subsequent modules to localization, like trajectory planning or path prediction, typically base their decisions on the latest estimated vehicle pose, whereas the vehicle's trajectory in the past is unused. This means that the output interface of our localization system only covers the latest poses $\boldsymbol{x}_\tau^p$ of each algorithm cycle. Estimating the trajectory is, however, a crucial factor for the accuracy of the graph's latest pose. This stems from the fact that linearization errors are reduced, and past data association decisions can easily be changed within the sliding window. As an overview, we illustrate the benefits w.r.t. data association in Figure 4.4.

A further important aspect of our approach is how we remove measurements from our sliding window graph. In the remainder of this chapter, we focus on pure localization and use truncation for removing outdated measurements from our sliding window graph. Compared to marginalizing out measurements, this means that we completely forget outdated measurements. While a disadvantage of truncation is that the end of the graph is less constrained compared to marginalization and we forget information, an advantage is that linearization errors can not accumulate over time. Also, truncation does not produce fill-in in the system matrix H and is therefore computationally faster. Truncation is a suitable approach as long as the sliding window contains sufficiently enough measurements to fully constrain the trajectory, the covered graph time span is long enough such that measurements at the end of the graph only have a marginal influence on the latest vehicle pose, and we are only interested in the latest vehicle pose. We will show in our evaluation that truncation is sufficient for our pure localization use case in urban areas. Nevertheless, using marginalization is beneficial as soon as we are interested in the estimated landmark positions for map refinement. We discuss marginalization in more detail and introduce an approach for mitigating its disadvantages for our use case in Chapter 5. In the following sections, we present the details of our graph-based sliding window approach.

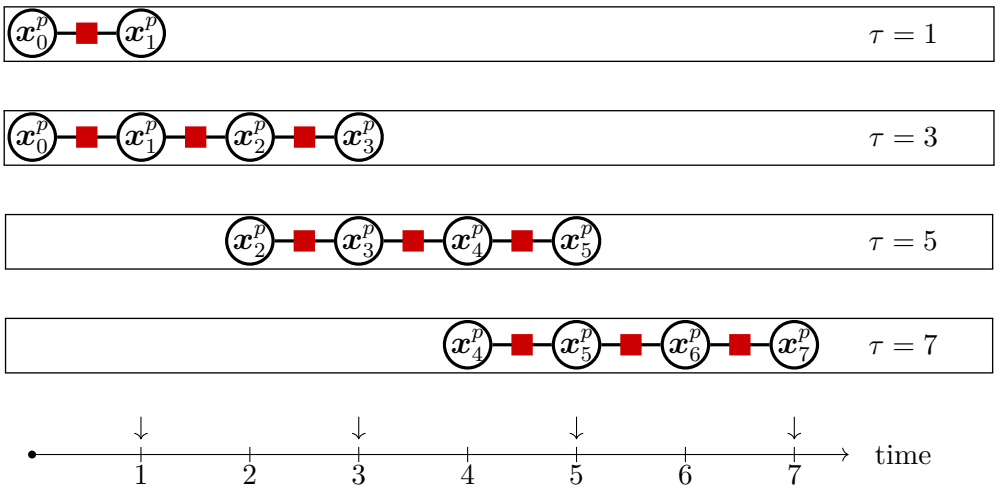
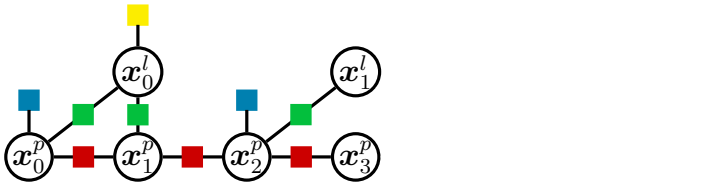
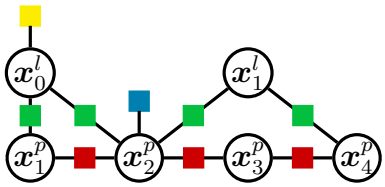


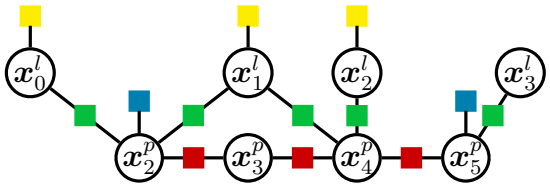
Figure 4.3: Example of a sliding window graph with an internal pose frequency of $f = 1\,\text{Hz}$ and a maximum number of poses $N = 4$. Instead of using marginalization for removing data, we truncate the sliding window graph in this example. We set the optimization frequency in this example to $f^o = 0.5\,\text{Hz}$, i.e., the algorithm cycle is executed every 2 s. For simplification, we here only illustrate the odometry factors and omit other factor types. The sliding window graph at time ($\tau = 1$) is still in its initialization phase as the maximum number of poses has not been reached yet. Starting at time ($\tau = 5$), each algorithm cycle removes two outdated poses and adds two new ones. The marks above the timeline at the bottom of the image illustrate when the algorithm cycle is executed.



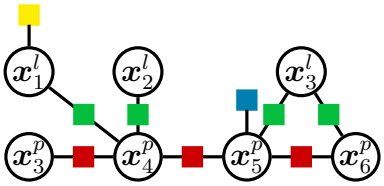
(a) $\tau = 3$



(b) $\tau = 4$



(c) $\tau = 5$



(d) $\tau = 6$

Figure 4.4: The figure illustrates our sliding window graph concept together with the benefits of modifying the structure of the factor graph. It serves as an introductory overview of the concepts that benefit from using sliding window graphs. In this example, we rely on truncation for removing measurements from our sliding window graph. We individually discuss *delayed measurements*, *delayed data association*, and *reversible data association* later in this chapter. The different subfigures represent the graph at different time steps τ. The maximum number of poses in this example is $N = 4$. In each timestep, the oldest pose is removed, and a new one is added. (a) Initial graph after the maximum number of poses has been reached. (b) Integration of a delayed landmark measurement that connects $\boldsymbol{x}_2^p$ and $\boldsymbol{x}_0^l$ (see Section 4.4.5). (c) Delayed map association of $\boldsymbol{x}_1^l$. This happens if, e.g., the map match previously was not probable enough (see Section 4.4.6). (d) The map match from $\boldsymbol{x}_2^l$ is considered wrong and reversed (see Section 4.4.7).

4.2.2 Optimization-based pose estimation

Our sliding window problem formulation follows the general notation for optimization problems presented in Section 3.4.1. While we are mainly interested in the vehicle pose, we simultaneously estimate the location of landmarks to compute map update hypotheses, as we will discuss in Chapter 5. In particular, we define the state optimization problem for a specific sliding window at time τ as

$$\boldsymbol{x}_\tau^* = \underset{\boldsymbol{x}_\tau}{\operatorname{argmax}}\, p(\boldsymbol{x}_\tau \mid \boldsymbol{z}^{\mathrm{lm}}, \boldsymbol{z}^{\mathrm{odo}}, \boldsymbol{z}^{\mathrm{abs}}, \boldsymbol{m}), \tag{4.1}$$

with landmark observations $\boldsymbol{z}^{\mathrm{lm}}$, odometry measurements $\boldsymbol{z}^{\mathrm{odo}}$, absolute pose measurements $\boldsymbol{z}^{\mathrm{abs}}$ (e.g., GNSS), and a map $\boldsymbol{m}$. For notational convenience, we will use $\boldsymbol{x} \equiv \boldsymbol{x}_\tau$ and $\boldsymbol{z} \equiv \boldsymbol{z}_\tau$ in the following, such that we can reuse the subscript for other purposes if necessary. Assuming i.i.d. measurements, Gaussian distributions (see Chapter 3.4.1) and outliers (see Chapter 3.4.4), we transform Equation (4.1) into

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{k}^{K} \rho\left(\boldsymbol{e}_k^\top \Omega_k \boldsymbol{e}_k\right),$$

which contains all involved constraints K that we select during graph construction for our current sliding window τ. For notational clarity, we omit the robust cost function ρ in the following and distinguish between the different types of constraints, which yields

$$\begin{aligned}\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} & \sum_i \mathrm{e}^{\mathrm{odo}}(\boldsymbol{x}^p, \boldsymbol{z}_i^{\mathrm{odo}})^\top \Omega_i^{\mathrm{odo}} \mathrm{e}^{\mathrm{odo}}(\boldsymbol{x}^p, \boldsymbol{z}_i^{\mathrm{odo}}) \\ & + \sum_i \mathrm{e}^{\mathrm{abs}}(\boldsymbol{x}^p, \boldsymbol{z}_i^{\mathrm{abs}})^\top \Omega_i^{\mathrm{lm}} \mathrm{e}^{\mathrm{abs}}(\boldsymbol{x}^p, \boldsymbol{z}_i^{\mathrm{abs}}) \\ & + \sum_i \mathrm{e}^{\mathrm{lm}}(\boldsymbol{x}^p, \boldsymbol{x}^l, \boldsymbol{z}_i^{\mathrm{lm}})^\top \Omega_i^{\mathrm{abs}} \mathrm{e}^{\mathrm{lm}}(\boldsymbol{x}^p, \boldsymbol{x}^l, \boldsymbol{z}_i^{\mathrm{lm}}) \\ & + \mathrm{F}^{\mathrm{map}}\left(\boldsymbol{x}^l, \boldsymbol{m}\right)\end{aligned} \tag{4.2}$$

with the error functions for odometry $\cdot^{\mathrm{odo}}$, absolute measurements $\cdot^{\mathrm{abs}}$, landmark measurements $\cdot^{\mathrm{lm}}$, and map association costs $\mathrm{F}^{\mathrm{map}}\left(\boldsymbol{x}^l, \boldsymbol{m}\right)$. We explicitly discuss the latter in more detail in Section 4.3.2, as the integration of map landmarks is fundamental to our approach. In our case, we use the Cauchy kernel as robust cost function as explained in Chapter 3.4.4. Additionally, we carefully select the used constraints in the first place, which we discuss in Section 4.4. We rely on the Gauss-Newton algorithm, as presented in Section 3.4.2, to minimize the cost function Equation (4.2).

4.3 Using third-party landmark maps

In the following, we present our approach for integrating general-purpose third-party landmark maps into our graph-based sliding window approach. We start by discussing third-party maps in Section 4.3.1. Afterward, we propose how to integrate third-party landmarks as priors into our state estimation in Section 4.3.2, how to estimate missing covariances in Section 4.3.3, discuss the impact on Gauss-Newton in Section 4.3.4, and consider landmarks with angular state components in Section 4.3.5.

4.3.1 Third-party maps

Using third-party maps imposes unique challenges for localization systems, which we discuss in the following. The process in which a map is created by a third-party is usually unknown. Hence it can be considered a black box. It might involve manually creating landmarks based on globally referenced aerial images or point clouds, as well as semi-automated or automated extraction from the available data. A key aspect of creating such a black box map is that the map elements must be considered as generic as possible. As the sensors and detectors used in a localization system can vary, the map landmarks should be general-purpose, i.e., detectable with different sensors. Thus, we require that the elements of a landmark map must be similar to our definition of a landmark for automated driving in Section 3.2. The opposite of a general-purpose third-party map is a map that is created with the same sensors, sensor mountings, and landmark detectors as later used for localization. On the one side an advantage of such a tailored map is that a vehicle with the same setup should be able to detect all mapped landmarks at runtime which eases map matching. On the other side a limitation is that the map might not be usable with other sensor types, e.g., because detection ranges are different. Thus, tailored maps have a limited transferability. This loss of versatility is mostly a disadvantage concerning cost-effectiveness and maintainability. Whenever updating a map is required, each tailored map must be changed individually. Whereas in the case of a generic landmark map updating the map must only be done once. The main challenge of relying on a general-purpose map is that the map elements may differ from what the detectors can observe at runtime. While the reasons for not observing a landmark can be sensor noise or blocked line-of-sight, it is also possible that detecting a general-purpose map landmark might not be possible with a specific landmark detector and sensor setup. For example, a landmark detector based on a LiDAR sensor with limited resolution might not be able to detect pole landmarks with a small radius, which, however, are part of the map. Undoubtedly there must be at least some overlap between what the vehicle is able to detect

and the map elements. Only then map matching is possible and the map is useful for localization. As a consequence, measurements of landmarks that are not part of the map must not be erroneously associated with the map. This might lead to localization failure. We consider these challenges within our data association and discuss them in more detail in Section 4.4. In addition to having a general-purpose landmark map that is usable with different sensor setups, our decision to rely on landmark maps has further reasons. At first, the data size of landmark maps is significantly smaller than storing raw sensor data or dense feature maps. Secondly, landmarks are easier to interpret by humans, which is vital for manually creating, updating, and especially verifying maps. Although this might be relaxed in the future due to increased automation within map creation and verification, it is an important point that eases the traceability of localization in case of localization failure and thus contributes to functional safety aspects. Figure 4.5 illustrates an excerpt of a general-purpose landmark map that we use within this thesis. We develop our data association concepts to consider the mentioned challenges. Nevertheless, our concepts also work with maps that are created for specific sensors. We do not impose the restriction of only using general-purpose maps but open up the possibility to do so.

Figure 4.5: An example of a general-purpose landmark map as used in this thesis. The figure shows a real-world excerpt of an urban area.

4.3.2 Integrating map factors as state priors

Our approach for utilizing landmark maps in graph-based localization is to incorporate map matches as priors over landmark states. For now, we assume that the data association is given and discuss the detailed aspects later in Section 4.4. Here we focus on the mathematical aspects of incorporating landmarks from a given map into graph-based optimization. We show how we define the map factors of our graph-based approach, as illustrated in Figure 4.6.

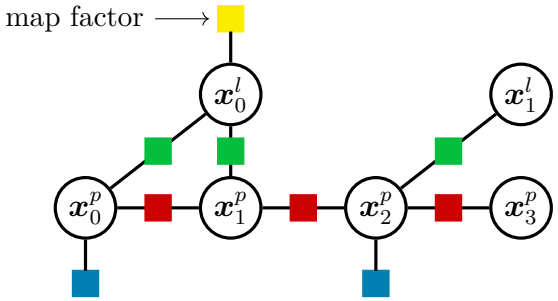


Figure 4.6: Exemplary sliding window graph that contains four vehicle poses $\boldsymbol{x}^p$ and two landmarks $\boldsymbol{x}^l$. The landmark $\boldsymbol{x}_0^l$ is constrained by a map factor.

We model each map association that we found during data association as a gaussian prior

$$p(\boldsymbol{x}_k^l) = \mathcal{N}\left(\boldsymbol{x}_k^l \mid \boldsymbol{m}_k, \Omega_k^{\text{map}}\right),$$

with the individual landmark state $\boldsymbol{x}_k^l \in \boldsymbol{x}^l$, the matched state of the map landmark $\boldsymbol{m}_k$, and its information matrix Ω_k^{map}. We use the index $k \in [1, \ldots, K]$ to identify individual matches within the total number of K map matches in the graph. Following our derivation in Section 3.4.1, we transform each prior into a cost function, such that the total map association costs are

$$\mathrm{F}^{\text{map}}\left(\boldsymbol{x}^l, \boldsymbol{m}\right) = \sum_k \mathrm{e}^{\text{map}}(\boldsymbol{x}_k^l, \boldsymbol{m}_k)^\top \Omega_k^{\text{map}} \mathrm{e}^{\text{map}}(\boldsymbol{x}_k^l, \boldsymbol{m}_k), \tag{4.3}$$

where we define the error function $\mathrm{e}^{\text{map}}(\boldsymbol{x}^l, \boldsymbol{m}_k)$ to match our error function notation. We define the error function itself as

$$\mathrm{e}^{\text{map}}(\boldsymbol{x}_k^l, \boldsymbol{m}_k) = \boldsymbol{x}_k^l - \boldsymbol{m}_k. \tag{4.4}$$

Note that the error function is valid whenever the underlying landmark state space is Euclidean, which is the case if the landmark is only defined by its position. If a landmark state contains angular information, like the orientation of a building corner, it is necessary to ensure that all operations produce valid results in the defined parameter space. We discuss this in Section 4.3.5.

4.3.3 Map covariance

A challenge of using third-party maps exists whenever covariance estimates for the map landmark states are neither available nor reliable. In that case, an educated guess for the covariance or information matrix Ω_k^{map} must be made. In addition to the i.i.d. assumption, we make the following assumptions to make an educated guess for the information matrix of map factors:

Assumption 4. *The position uncertainty of any landmark in the general-purpose map position is assumed to be isotropic.*

Assumption 5. *All third-party map landmarks of the same type share the same covariance estimate.*

In our case, Assumption 5 is required because no distinct covariances are available from the map provider. If distinct and meaningful covariances are available Assumption 5 can be omitted. Nevertheless, an advantage of assuming shared covariance estimates is that it reduces the required computations, as we will show in the next section. Following Assumption 4 and the i.i.d. assumption, we write the information matrix as

$$\Omega_k^{\text{map}} = \omega_k^{\text{map}} \mathbf{I},$$

with the map information factor ω_k^{map}. Taking into account Assumption 5, we rewrite the equation as

$$\Omega_k^{\text{map}} = \omega^{\text{map}} \mathbf{I}, \tag{4.5}$$

where now ω^{map} is shared between all landmarks. For notational convenience, we omit that the covariance scalar is not necessarily equal for all landmark types. To make an educated guess for the map information factor ω^{map}, we suggest the following heuristic. Instead of directly using the map information factor, we understand it as more graspable to estimate the covariance or the standard deviation such that we can use it as a parameter to our approach. Our notation and the conversion to the map information factor is

$$\omega^{\text{map}} = \frac{1}{\sigma^2},$$

with the covariance factor σ^2. Following this, we estimate the covariance factor as

$$\sigma^2 = \frac{1}{\gamma(c)} r^2, \tag{4.6}$$

where γ is the two-dimensional inverse-chi-squared cumulative distribution function, c the confidence interval, and r a user-defined radius. Assuming that, e.g., 95% ($c = 0.95$) of all map landmarks were measured within an error radius of $r = 0.02\,\text{m}$, the educated covariance guess with Equation (4.6) results in $\sigma^2 \approx 6.7 \times 10^{-5}$. This example illustrates how we guess the map covariance after manual inspection if no further details about the map quality are available from the map provider.

4.3.4 Impact on Gauss-Newton

Our assumptions about map landmarks allow us to simplify the involved terms in Gauss-Newton. We first note that we can rewrite the map cost function from Equation (4.3) as

$$\mathrm{F}^{\mathrm{map}}\left(\boldsymbol{x}^l, \boldsymbol{m}\right) = \omega^{\mathrm{map}} \sum_k \mathrm{e}^{\mathrm{map}}(\boldsymbol{x}_k^l, \boldsymbol{m}_k)^\top \mathrm{e}^{\mathrm{map}}(\boldsymbol{x}_k^l, \boldsymbol{m}_k),$$

where we have used Equation (4.5).

Following the fact that our map error function from Equation (4.4) is linear, we write its Jacobian as

$$\mathrm{J}_k(\breve{\boldsymbol{x}}_k^l) = \mathbf{I},$$

where $\breve{\boldsymbol{x}}_k^l$ denotes the linearization point and $\mathbf{I}$ the identity matrix. In return, we can simplify the computation of the system matrix H and gradient vector $\boldsymbol{b}$ involved in Gauss-Newton. We rewrite the summands from Equation (3.7) that correspond to the map costs as

$$\begin{aligned} H &= H_{\setminus k} + \sum_k \mathrm{J}_k(\breve{\boldsymbol{x}}_k^l)^\top \Omega_k^{\mathrm{map}} \mathrm{J}_k(\breve{\boldsymbol{x}}_k^l) = H_{\setminus k} + \omega^{\mathrm{map}} K \mathbf{I}, \\ \boldsymbol{b} &= \boldsymbol{b}_{\setminus k} + \sum_k \mathrm{e}^{\mathrm{map}}(\breve{\boldsymbol{x}}_k^l, \boldsymbol{m}_k)^\top \Omega_k^{\mathrm{map}} \mathrm{J}_k(\breve{\boldsymbol{x}}_k^l) = \boldsymbol{b}_{\setminus k} + \omega^{\mathrm{map}} \sum_k \mathrm{e}^{\mathrm{map}}(\breve{\boldsymbol{x}}_k^l, \boldsymbol{m}_k), \end{aligned}$$

where $H_{\setminus k}$ and $\boldsymbol{b}_{\setminus k}$ denote the corresponding parts without any map factors. For notational convenience, we ignore the dimensional mismatch between the summands. It must be ensured that the summands are added to the correct rows and columns of H and $\boldsymbol{b}$.

4.3.5 Map landmarks with angular state components

To handle landmark states with angular components, we rewrite the map error function from Equation (4.4) in the more general form

$$\mathrm{e}^{\mathrm{map}}(\boldsymbol{x}_k^l, \boldsymbol{m}_k) = \boldsymbol{x}_k^l \boxminus \boldsymbol{m}_k,$$

with $\boxminus$ denoting a manifold operation. While in our 2D case the manifold operation corresponds to the normalization of the involved angles after subtraction, it is more involved in the 3D case as singularities must be considered. For the latter, we refer to Grisetti et al. (2010) for a more detailed discussion and to Hertzberg (2008) for more mathematical details on manifolds. In addition to using manifold operations within the error functions, the Taylor approximation and state update equations in the Gauss-Newton optimization are affected accordingly as we have shown in Section 3.4.3.

Regarding the map information factor, we so far omitted that the covariances for angular elements must be handled separately. The corresponding derivation can be done similar to the derivations without angles. For example, in the case of building corners, which we do not further consider in this thesis, the information matrix for a map landmark could be

$$\Omega^{\text{map}}_{k_{\text{corner}}} = \begin{bmatrix} \omega^{\text{map}}\mathbf{I}_{2x2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \omega^{\text{map}}_{\alpha} & 0 \\ \mathbf{0} & 0 & \omega^{\text{map}}_{\theta} \end{bmatrix},$$

where $\omega^{\text{map}}_{\alpha}$ denotes the map information factor for opening angles and $\omega^{\text{map}}_{\theta}$ the one for orientation angles of building corners. Both must be guessed in addition to the overall map information factor ω^{map} for positions.

4.4 Data association

The main design challenge of our data association approach is to render using general-purpose landmark maps possible. Typical general-purpose maps for landmark-based localization contain semantic objects like lamp posts, road markings, reflector posts, and others depending on the environment scenario and use case. In more detail, we present our definition of a landmark in Section 3.2 and discuss third-party maps in general in Section 4.3.1. One of the main challenges we address in our approach is that the set of detected landmarks may only partially match the landmarks in the general-purpose map. On the one side, this is caused by missing map landmarks and false positive detections, whereas on the other side, the map contains landmarks that are not detectable by the vehicle at all. For example, a map may contain pole-like objects with varying diameters, in which case a vehicle that is only equipped with a low-resolution LiDAR might never be able to detect the ones with small diameters. Compared to using maps explicitly designed for a specific sensor, i.e., detectable and map landmarks are well aligned, we prefer having a general-purpose map that can be used with a variety of sensors. We consider this advantageous over creating individual maps for specific sensor types. It increases the transferability and maintainability of the used map and such may heavily reduce costs.

Our approach to data association is to distinguish between three subsequent parts, which we call *local association*, *map matching*, and *temporal association smoothing*. We find out if multiple measurements belong to the same landmark within local association, whereas in map matching, we match the identified landmarks to a given map. A crucial aspect of our map matching is that it only suggests map matches for the current sliding window, which happens independently from previous matching results. Instead, we track the suggested matches

over time in our temporal association smoothing step and identify the most probable associations for all individual landmarks. This helps us to robustify our approach against outliers and resolve ambiguous map matches over time. We explain the different steps of our data association approach in the following.

4.4.1 Local association

Our first step in graph construction is to find out if measurements within our sliding window belong to the same landmark. Following our design principles, our approach does not rely on any feedback but instead is independently solved upfront to map matching and graph optimization. Within our approach, we first project all raw landmark measurements within a sliding window's time span into a common local coordinate system based on accumulated odometry data. Afterward, we spatially group the projected measurements into *landmark clusters*. In our algorithm, we iteratively loop over all projected measurements and assign them individually to either an existing landmark measurement cluster or a new one. In more detail, we compute the distance between a projected measurement and the center positions of surrounding clusters and decide based on a threshold if they belong together. Whenever we assign a landmark measurement to a cluster, we fuse the projected measurement into the cluster and update the cluster's center position. We illustrate the local coordinate system and projected landmark measurements in Figure 4.7. Algorithm 2 gives an overview of the different steps involved in solving the local association problem. We apply this algorithm independently for each landmark type. We denote the set of landmark clusters as $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_Q\}$, where Q is the initially unknown number of clusters. During local association, each landmark measurement is assigned to only one cluster.

Our local association produces two different outputs. First, the decision about which landmark measurements belong together, such that we can create the factors between vehicle poses and landmark states in our sliding window graph. Second, the cluster centers which represent a local map that we use during map matching. Our local association only influences the structure of the graph, while the original measurements remain untouched.

An important aspect of our local association algorithm is that it requires a near drift-free odometry as input data. The negative effect of odometry drift within our local association is that the landmarks measurements are inaccurately projected into the local coordinate system. The more the odometry drifts, the less accurate is the projection. As a consequence, landmark measurements might falsely get clustered during local association. Also, the local map that is created during local association and used for map matching gets distorted and thus negatively influences our map matching step. Therefore, we make the following assumption.

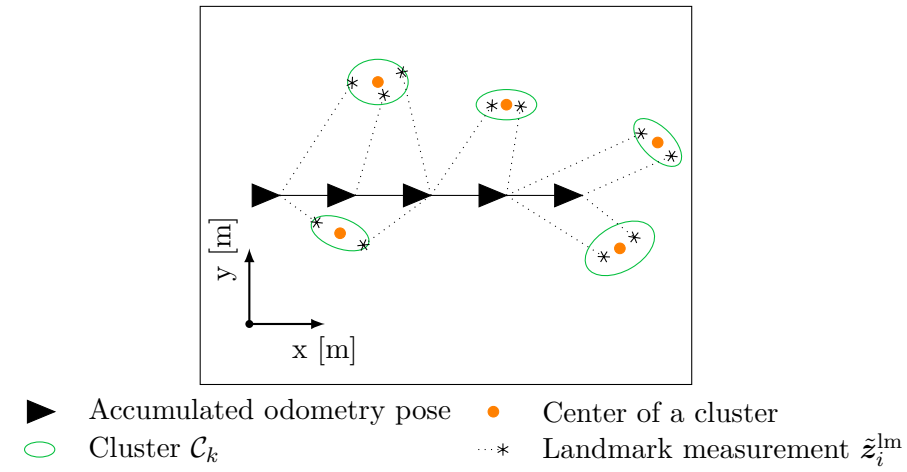


Figure 4.7: Illustration of our local association. We first accumulate odometry measurements, which we use to project landmark measurements from VRF into a common local coordinate system. Afterward, we identify clusters of landmark measurements. The results of our local association step are the cluster centers, which we use for map matching, and the assignments of measurements to clusters that we use during graph construction.

Assumption 6. *The accumulated odometry drift within the time span relevant to local association is neglectable.*

In practice, this limits the length of the sliding window that can be used for local association because the accumulated drift increases with an increasing sliding window length.

An important parameter of our Algorithm 2 is the distance threshold $d^c_{\max}$. We associate a measurement to the closest cluster whenever their distance in the local coordinate system is below the threshold $d^c_{\max}$. Considering that landmark measurements are noisy, the parameter must be tuned such that measurements are correctly clustered together. There are two distinct error types that we need to consider. If the threshold is too high, measurements might falsely get clustered together. Whereas if the threshold is too low, the measurements might falsely get split into multiple clusters. In our case, the latter error type is favorable, such that we suggest choosing a value that rather is too low than too high. This is because we allow multiple landmark measurement clusters to be associated with the same map landmark during map matching. In this case, the corresponding landmark clusters have the same map prior during graph optimization, which affects the vehicle's trajectory as if the landmark clusters were correct.

However, even if measurements are falsely clustered during local association, our localization approach is still robust against these outliers to some extend. We achieve this by using robust kernel functions, like Cauchy, during the graph optimization phase. The kernel functions decrease the negative impact of false cluster associations by limiting the incorporated error (see Section 3.4.4). In

return, our localization result is likely to be still valid even if some measurements are associated to a wrong cluster.

The applicable distance metric for $d^c_{\max}$ depends on the use case and the landmark type. The simplest option is the Euclidean distance between the current cluster center and the measurement, which, e.g., is applicable for pole measurements. If the landmark measurements contain reliable covariance estimates, a more sophisticated alternative is the Mahalanobis distance, which is preferable as it takes the covariance of the measurement into account. Compared to scan-matching approaches, e.g., iterative closest point (ICP)-based, our odometry-based algorithm reliably works if the vehicle does not steadily detect all landmarks in every time step and even handles situations with very few and noisy detections.

Our local association approach's main advantage is that it even works robustly in environmental scenarios with few landmarks. It is independent of the number of landmarks and even works with a single landmark detection, while its computational effort is minimal even in scenarios with many landmarks. Furthermore, it is independent of the landmark measurement frequency, measurement delay, and works with out-of-sequence measurements.

```
1  Function LocalAssociation(new landmark measurements z̃^lm_{τ+},
     accumulated odometry a, clusters C, distance threshold d^c_max):
2      extend accumulated odometry a using z̃^odo_{τ+};
3      foreach z̃^lm_i in z̃^lm_{τ+} do
4          project z̃^lm_i into local coordinate system using a;
5          find closest cluster C_k in C with distance < d^c_max;
6          if found C_k then
7              create new cluster with z̃^lm_i;
8              add new cluster to C;
9          else
10             add z̃^lm_i to cluster C_k;
11             recompute cluster center of C_k;
12     return clusters C;
```

Algorithm 2: Pseudo code for our local association strategy. Our local association algorithm iteratively associates landmark measurements to either existing landmark measurement clusters or creates new ones. We execute this algorithm once in every main algorithm cycle.

4.4.2 Map Matching

The next step after local association in our architecture is map matching. We use map matching to find the best overlap between our general-purpose map and the landmarks that the vehicle perceives. In combination with our temporal association smoothing, the found map matches allow us to globally constrain the landmark positions in our sliding window graph as explained in Section 4.3.2. At first, we take all landmark clusters from local association and estimate the mean c of each cluster $\mathcal{C}$ as its landmark position in the local map. We represent our local map as

$$l = \{c_1, \ldots, c_Q\},$$

with Q landmarks in the local map. This local map contains all landmarks that were detected by the vehicle in the current sliding window. Map matching in our case is the problem of finding a transformation matrix that optimally projects the local map onto the global general-purpose map. Figure 4.9a illustrates an example in which the landmarks of the local map are denoted as orange cluster centers, while Figure 4.9b shows an example of a global map. As the scale of the local map is already accurate, the map matching problem reduces to finding the translational and rotational components of an affine transformation matrix. More specifically, we compute a transformation matrix independently for each sliding window τ, which gives us the best transformation matrix estimate T_τ^*. We denote a set of globally projected landmarks from our local map as

$$l_T = \{c_{T1}, \ldots, c_{TQ}\},$$

where l_T is now the set of globally projected landmarks from local association, T is some transformation matrix, and c_{Ti} are the projected individual landmarks. For visual reference, Figure 4.10 illustrates an example projection from our local map into global coordinates. Following our design principles, it is important to note that the transformation matrix T is independent of any graph optimization. It only depends on the global map and the landmark clusters derived from the raw odometry and landmark measurements. We write map matching as the cost minimization problem

$$T_\tau^* = \underset{T}{\operatorname{argmin}}\, c\left(l_T, g\right), \tag{4.7}$$

where $c\left(l_T, g\right)$ is the cost function between a transformed local map l_T and global map g.

An essential challenge that we consider in our approach is that the general-purpose map elements are not necessarily tuned to reflect what the vehicle is actually able to detect. This implies that the map matching algorithm must

consider that the overlap of the local and global map might be limited. Therefore, we design our cost function $c(l_T, g)$ from Equation (4.7) in a way that allows us to match as many local landmarks to the map as possible while simultaneously making sure that the distances for all matches are minimal. We approximate this with the cost function

$$c(l_T, g) = \sum_{c \in l_T}^{Q} f(c, g), \tag{4.8}$$

$$f(c, g) = \begin{cases} d_\mathrm{c} & d_\mathrm{c} < d_\mathrm{max} \\ d_\mathrm{max}\eta & d_\mathrm{c} \geq d_\mathrm{max} \end{cases}, \tag{4.9}$$

$$d_\mathrm{c} = \min_g d(c, g),$$

where f is the cost that is assigned to each landmark cluster c, d_c is the distance to the closest map landmark for cluster c, η is a weighting parameter, and d_max is the maximum distance threshold that decides if a cluster and a map match can be considered matches. The latter threshold is only used within our map matching. It does not decide whether the match is included in the graph or not. The applicable value of d_max depends on how accurate the map and the landmark detections are. In the scope of this thesis, we use $d_\mathrm{max} = 1\,\mathrm{m}$ for, e.g., pole landmarks.

The key aspect of our matching approach is Equation (4.8). It evaluates the applied transformation matrix T by favoring well-fitting matches and punishing non-matches. If a landmark from the local map is matched by the criteria described above, it contributes to the cost function with its distance to the map. Otherwise, the maximum allowed distance is multiplied by the weighting parameter η. The weighting parameter ensures that we match as many landmarks as possible instead of selecting a few good ones. This is especially important for general-purpose landmark maps containing many landmarks that the vehicle can not detect. Together with measurement noise and false positive landmark detections, it is likely in some situations that a small set of landmarks erroneously seem to fit well, which is why it is crucial to match as many landmarks as possible. We discuss the distance threshold d_max and the role of the weighting parameter η in more detail in Section 4.4.3.

Since we rely on landmarks, which are naturally sparser than, e.g., raw sensor measurements, and limit ourselves to the 2D case, it is possible to search for map associations by solving Equation (4.7) in a brute force manner. In brief, our approach reduces the map matching problem to creating a list of transformation matrix candidates and evaluating them individually. We then pick the best transformation matrix that produces the lowest sum of distances for all associated and unassociated landmarks. We provide a pseudo code implementation

of Equation (4.7) in Algorithm 3, visualize the essential steps from Figure 4.9 to Figure 4.14, and describe our algorithm in the following in more detail. Our algorithm's input is the local map from local association and our general-purpose map in global coordinates, which we illustrate in Figure 4.9. Additionally, our algorithm requires an initial vehicle pose (e.g., from low-cost GNSS), the discussed maximum distance threshold $d_{\max}$, and the maximum transformation vector length $d_{T_{\max}}$ that limits the search radius. We start by projecting our local map into global coordinates based on the initial vehicle pose, as shown in Figure 4.9. Afterward, we create a list of potential transformation matrices for each landmark from the local map based on all map landmarks within the specified radius $d_{T_{\max}}$, as illustrated in Figure 4.11. In this step, we consider transformation matrices with rotational components by rotating the local map around the initial vehicle pose and performing the illustrated transformation candidate generation for each potential rotation. Next, we loop through all potential transformation matrices, temporarily apply each transformation matrix to the local map to calculate its cost (Equation 4.8). Figure 4.12 and Figure 4.13 illustrate examples of applying a transformation matrix and the individual costs for each projected and transformed cluster. In total, Table 4.1 shows the cost table for all transformation matrices in our example from which we choose the one with the lowest cost as our optimal transformation matrix T_τ^* in the current algorithm cycle. In comparison to ICP-based algorithms, which typically require a good initial guess and are prone to get stuck in local minima (Pomerleau et al., 2015), our approach only requires a rough initial pose estimate within the maximum search radius $d_{T_{\max}}$ and yields the best transformation matrix with the global cost minimum.

After finding the best transformation matrix, we compute the map matches that best fit the measurement data in the current sliding window, as shown in Figure 4.14. Applying transformation matrix T_τ^* to the local map and searching for the closest map landmark gives us the set of map matches $\mathcal{M}_\tau$ for the current sliding window τ, which is the input for our temporal association smoothing, which we will explain in Section 4.4.4.

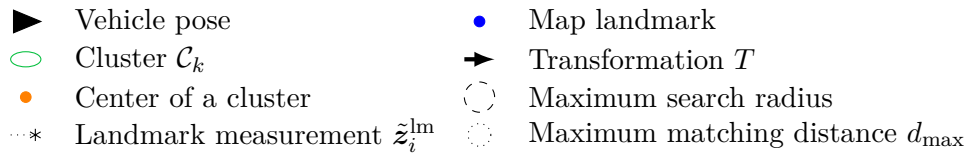
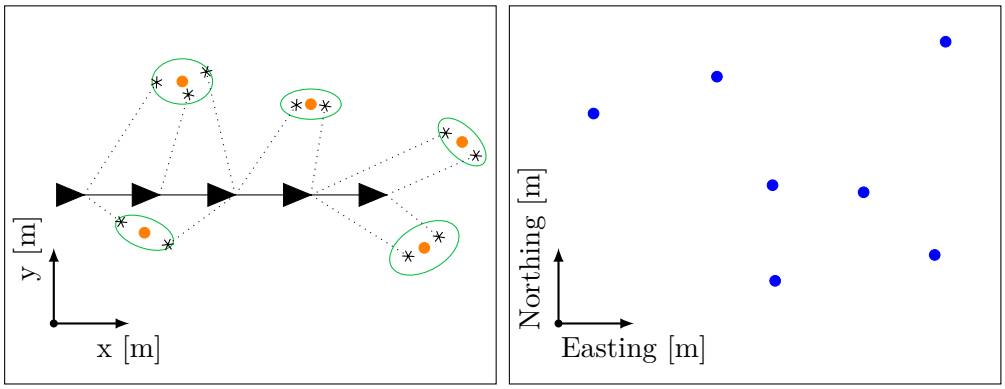
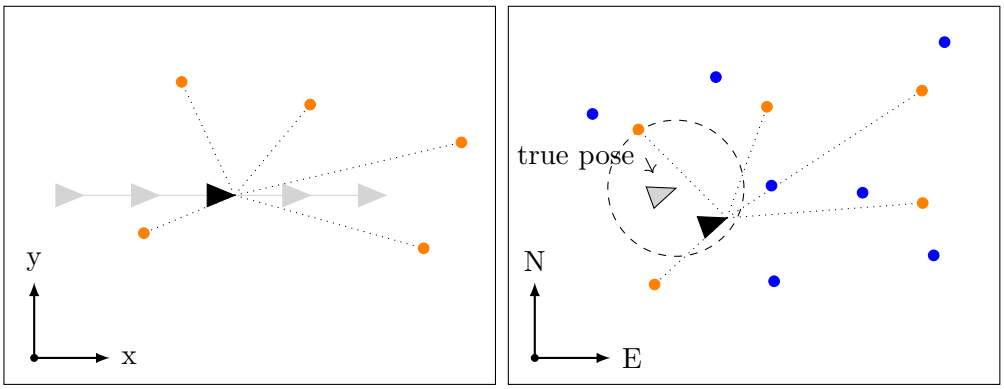


Figure 4.8: Legend for the subsequent Figures 4.9 to 4.14 in this section.



(a) Local association (b) Global landmark map

Figure 4.9: Example of our local association in (a) and the corresponding landmark map in global UTM coordinates in (b). Within our local association, we accumulate odometry measurements to create a pose for each unique timestamp of all landmark measurements inside our sliding window. We attach each landmark measurement to its corresponding local pose to compute its position inside the local map. This allows us to obtain a set of clusters, such that we have identified which measurements belong to the same landmark. The cluster centers, which are shown in orange, represent the local map l that we want to match to the global landmark map.



(a) Before projection (b) After projection

Figure 4.10: The figure illustrates how we project the local map onto the global coordinate system. First, we calculate the positions of all cluster centers relative to one vehicle pose, as shown in (a). Afterward, we project the local map into the global coordinate system based on an initial global vehicle pose, which we draw in black in (b). The initial vehicle pose is either the last received GNSS pose or a pose estimate from previous algorithm cycles of our approach. We assume that the initial pose is within a predefined radius around the true unknown vehicle pose. This is based on the maximum search radius $d_{\max}$ during our transformation matrix search, which we illustrate in Figure 4.11. In our case, we use $d_{\max} = 10\,\mathrm{m}$.

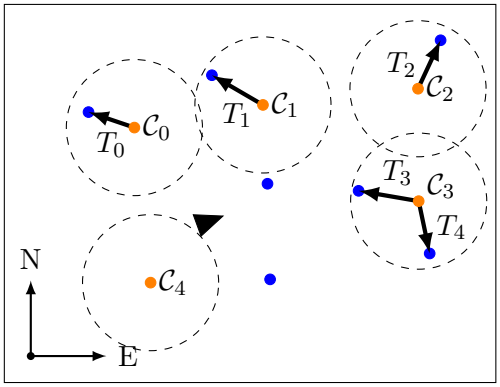
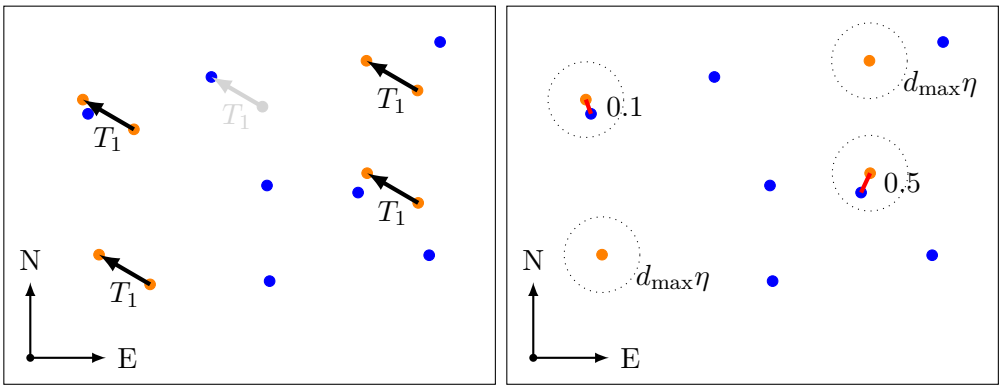
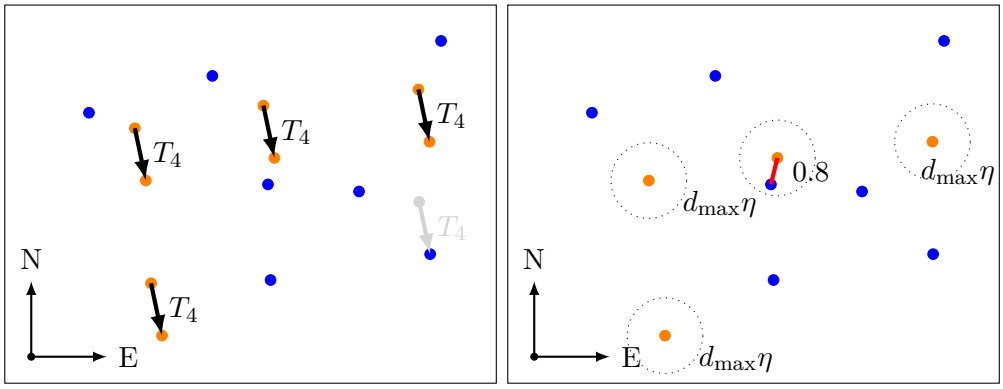


Figure 4.11: The figure illustrates how we generate a list of transformation candidates. Based on the initial projection (shown in Figure 4.10b), we consider for each cluster from the local map all map landmarks within a specified radius $d_{\max}$ as possible matches and create a corresponding transformation T as a candidate. We individually evaluate all identified possible transformation matrices, which we exemplarily illustrate in Figure 4.12 and Figure 4.13. The results are shown in Table 4.1. For clarity, we limit our example to translation only and exclude any transformation matrices with rotational components.



(a) (b)

Figure 4.12: This figure illustrates how we apply a transformation candidate and evaluate it. In this example, we examine the transformation T_1 from Figure 4.11. (a) We apply the transformation T_1 to all clusters except the one in gray, which was used to generate the transformation. (b) We evaluate the transformation by computing cost terms for each transformed cluster. If a transformed cluster has no map landmark within the maximum matching distance $d_{\max}$, its corresponding cost is $d_{\max}\eta$. Otherwise, its cost is the Euclidean distance to the closest map landmark. We explain the weighting parameter in Section 4.4.3. The costs for all transformations of our example are given in Table 4.1.



(a) (b)

Figure 4.13: This figure is similar to Figure 4.12 except that we here illustrate the transformation T_4. In this example, the transformation is likely wrong since only one cluster has a map landmark within the maximum matching radius, while all others can not be matched. The costs for all transformations of our example are given in Table 4.1.

	c_0	c_1	c_2	c_3	c_4	$\sum$
T_0	-	0.2	$d_{\max}\eta$	0.6	$d_{\max}\eta$	8.8
(see Figure 4.12) T_1	0.1	-	$d_{\max}\eta$	0.5	$d_{\max}\eta$	**8.6**
T_2	0.5	$d_{\max}\eta$	-	$d_{\max}\eta$	$d_{\max}\eta$	12.5
T_3	0.15	0.5	$d_{\max}\eta$	-	$d_{\max}\eta$	8.65
(see Figure 4.13) T_4	$d_{\max}\eta$	0.8	$d_{\max}\eta$	$d_{\max}\eta$	-	12.8

Table 4.1: Cost table for our map matching example shown in Figures 4.9 to 4.14. The table summarizes the cost for all transformation candidates. In more detail, Figure 4.12 and Figure 4.13 illustrate how we compute the cluster costs c_k for the transformations T_1 and T_4. In our example, we use $d_{\max} = 1\,\text{m}$ and $\eta = 4$, such that $T_\tau^* = T_1$ is the best transformation with the lowest cost of 8.6. We use the best transformation to identify the set of map matches for the current sliding window, as shown in Figure 4.14.

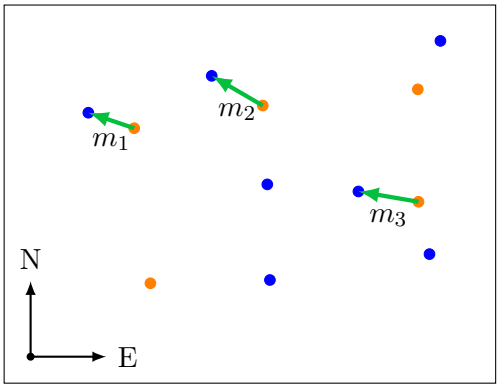


Figure 4.14: The figure illustrates the three map matches as green arrows that our algorithm identifies within our example. We obtain the set of map matches for the current sliding window $\mathcal{M}_\tau = \{m_1, m_2, m_3\}$ based on the best transformation T_τ^*, as stated in Table 4.1. In this step, we reuse the results from the previously applied transformation as shown in Figure 4.12b.

1 **Function** `FindBestTransformation`(local map l, global map g, pose $\boldsymbol{x}^p$, maximum matching distance $d_{\max}$, maximum transformation vector length $d_{T_{\max}}$)**:**

`/* Project local map to global frame. See Figure 4.10. */`

2 T_{init} = `GetApproximateTransformation(`$\boldsymbol{x}^p$`)`;

3 l_{Tinit} = `ApplyTransformation(`l, T_{init}`)`;

`/* Generate transformation matrix candidates. See Figure 4.11 */`

4 tCandidates = `GenerateCandidates(`l_{Tinit}, g`)`;

`/* Get potential map matches for all local landmarks */`

5 g_l = `GetMapLandmarksInRange(`l_{Tinit},g,$d_{T_{max}} + d_{max}$`)`;

`/* Initialize variables */`

6 minCostForBestTransformation = Inf;

7 T^*_τ = empty;

`/* Main loop */`

8 **foreach** T in tCandidates **do**

9 l_T = `ApplyTransformation(`l, T`)`;

10 costForT = 0;

11 **foreach** landmark $\boldsymbol{x}^l_{l_j}$ in l_T **do**

12 $d_{\min} = d_{\max}\eta$;

`/* Loop over all map landmarks that were previously identified as potential matches. */`

13 **foreach** landmark $\boldsymbol{x}^l_{g_k}$ in $g_l(l_j)$ **do**

14 d = `Distance(`$\boldsymbol{x}^l_{l_j}, \boldsymbol{x}^l_{g_k}$`)`;

15 **if** d $< d_{\max}$ AND d $< d_{\min}$ **then**

16 $d_{\min} = d$;

17 costForT += $d_{\min}$;

18 **if** costForT $<$ minCostForBestTransformation **then**

19 minCostForBestTransformation = costForT;

20 $T^*_\tau = T_i$;

21 **return** best transformation T^*_τ;

Algorithm 3: Pseudo code for our algorithm that finds the optimal transformation matrix T^*_τ between our local map l and the given map g. The algorithm is exemplarily illustrated in Figures 4.8 to 4.14.

4.4.3 Tuning the weighting parameter η

An aspect of our map matching algorithm is to apply various transformation matrices to a locally created map and find the one transformation matrix that produces the best overlay with the given map. Therefore, it is necessary to assess the overlay such that we can compare different transformation matrices to each other and find the one that produces the minimal cost, i.e., yields the best overlay with the map. When tuning the weighting parameter η in Equation (4.9), we need to consider two aspects that we explain in the following. Figure 4.15 provides an example of the cost function Equation (4.9) and illustrates different parameter choices. First, our map matching should prefer an overlay with many noisy

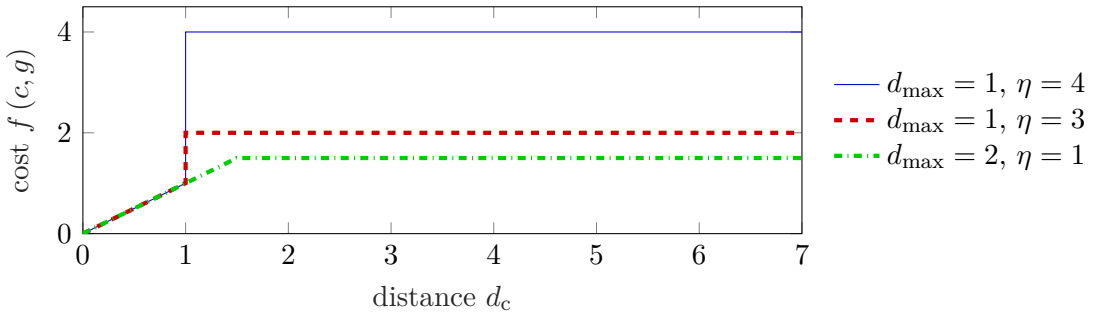
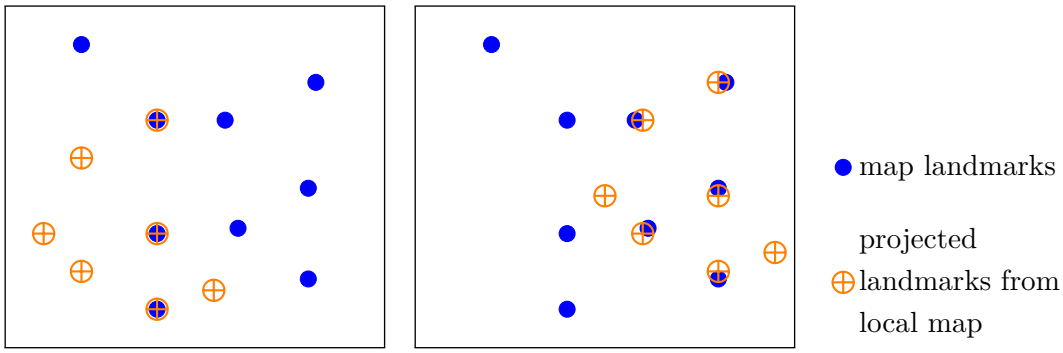


Figure 4.15: Examples of different parameter choices for the cost function in Equation (4.9) used to assess a matching between a detected and a map landmark. The distance d_c is the distance between a detected landmark in the transformed local map and its closest map landmark. In the case of pole landmarks, d_c corresponds to the Euclidean distance. Within this thesis we choose $d_{\max} = 1$ and $\eta = 4$.

matches over one that contains only a few perfect matches. This is necessary due to the inherent position noise in the local map. It ensures that the unity of all landmarks in the local map is considered and prevents that a few good but wrong matches are preferred. Figure 4.16 illustrates this issue. Within the cost assigned to a transformation matrix, the weighting parameter η serves as a punishment for not matching a landmark from the local map to the general-purpose map. As a consequence, our algorithm tries to match as many landmarks as possible.

Second, we need to consider that not all detected landmarks in the local map are part of the given general-purpose map. This might be caused by, e.g., false positive detections or an incomplete given map. Therefore, we assume that there are always landmarks that can not be matched to the given general-purpose map. Consequently, the weighting parameter η can not be chosen to be arbitrarily high but instead must be limited. It must be chosen in a way such that a few unmatched landmarks do not overrule an otherwise plausible overlay.

Both points contradict each other, such that we need to find a reasonable balance when setting the weighting parameter η. Figure 4.16 illustrates this issue.



(a) Example 1 based on transformation matrix T_A

(b) Example 2 based on transformation matrix T_B

Figure 4.16: An example of two boundary cases while assessing two different transformation matrices. The images (a) and (b) illustrate the result of applying two different transformation matrices. The configuration of the local landmarks is the same in both images, except that their projection onto the global map is different. Our map matching algorithm is designed to prefer Example 2, which we influence by setting the weighting parameter η. The corresponding costs for both examples are illustrated in Figure 4.17. (a) Three local landmarks perfectly match the global map, whereas four are not matched. (b) Five landmarks are in each case close to a map landmark and matched with the maximum allowed distance. Two landmarks are not matched.

To analyze the effect of the weighting parameter η, we investigate the upper and lower cost boundaries of Equation (4.8) together with the percentage of local landmarks that are matched to the map. We denote the latter as the matching ratio $\alpha \in [0, 1]$. A ratio of $\alpha = 1$ implies that all detected landmarks, i.e., the landmarks in the local map, can be matched to a map landmark, whereas in the case of $\alpha = 0$, none of the detected landmarks can be matched to the map. Given some matching ratio α, the lower bound of Equation (4.8) is reached if all of the matched landmarks perfectly overlay with their assigned map landmark. In this case, the cost of each map match is zero, such that the overall cost is defined by the remaining landmarks that could not be matched. We write this lower bound for the average matching cost per cluster landmark with respect to the matching rate α as

$$c_{\downarrow}(\alpha) = d_{\max}\eta(1 - \alpha). \tag{4.10}$$

Vice versa, the upper cost bound

$$c_{\uparrow}(\alpha) = d_{\max}\eta(1 - \alpha) + d_{\max}\alpha \tag{4.11}$$

is reached if all matched landmarks have a distance that is equal to the maximum allowed matching distance $d_{\max}$. We show both bounds in Figure 4.17, which illustrates the role of the weighting parameter η. It shows which boundary cases produce the same costs for a few exemplary weighting parameters η. Throughout this thesis, we choose to set the weighting parameter to $\eta = 4$.

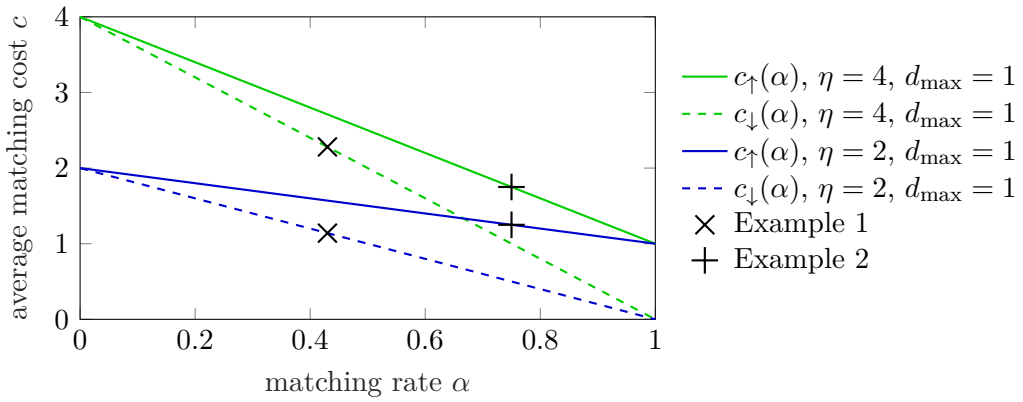


Figure 4.17: Impact of the weighting parameter η in our map matching algorithm. The figure shows the upper and lower bound average cost of Equation (4.8) depending on the map matching rate α for various choices of the weighting parameter η and matching threshold $d_{\max}$. The solid lines represent the upper boundaries $c_\uparrow(\alpha)$ (see Equation (4.11)), whereas the dashed lines show the lower boundaries $c_\downarrow(\alpha)$ (see Equation (4.10)). As a more specific application, the average cost for the two examples in Figure 4.16 is shown for two sets of parameters. $\eta = 4$, $d_{\max} = 1$ corresponds to our parameter choice within this thesis. In this case, the cost of Example 2 is lower compared to the cost of Example 1 and thus selected by our algorithm. Vice versa, for $\eta = 2, d_{\max} = 1$, the choice would be reversed.

4.4.4 Temporal association smoothing

In this step, we compute a temporally consistent set of map associations based on all map matches, i.e., detected landmarks matched to map landmarks found in previous algorithm cycles. In comparison, our map matching approach so far was only based on the measurement data within the current sliding window τ. By design, our graph-based sliding window evolves over time, which means we incorporate new measurements and discard old ones in every algorithm cycle. Hence, our map matching algorithm runs on different data in every algorithm cycle, which possibly changes the found map matches between subsequent algorithm cycles. By taking into account the matching results of previous sliding windows, we robustify our approach against outliers. Besides, by keeping track of all found matches, our *temporal association smoothing* helps to resolve ambiguous map matches over time. This is because our approach allows us to add the most probable map associations to the graph. We define the problem of finding the most probable map match for an individual landmark $\boldsymbol{x}_i^l$ based on the matching history as

$$m_i^* = \underset{m_i}{\operatorname{argmax}}\, p(m_i \mid \mathcal{M}_1, \ldots, \mathcal{M}_\tau), \tag{4.12}$$

where m_i^* is the map landmark that we matched most often to the detected landmark $\boldsymbol{x}_i^l$ while considering all map matching sets $\mathcal{M}$ of previous algorithm cycles. In this sense, we consider the map landmark m_i^* as the optimal choice

for the perceived landmark $\boldsymbol{x}_i^l$. This helps us overrule individual outliers for each landmark and joint outliers for all landmarks that might occur during our map matching step. Also, temporally smoothing the associations helps us to fill in matches if missed during map matching. In our approach, we accumulate the results of our map matching step and find the map landmark that we matched most often to a detected landmark. We refine Equation (4.12) as

$$\begin{aligned} m_i^* &= \underset{m_i}{\operatorname{argmax}} \frac{\sum_t \rho(\mathcal{M}_t, m_i)}{\sum_t \sum_j \rho(\mathcal{M}_t, m_j)}, \\ &\propto \underset{m_i}{\operatorname{argmax}} \sum_t \rho(\mathcal{M}_t, m_i) \\ \rho(\mathcal{M}_t, m_i) &= \begin{cases} 1 & m_i \text{ is matched to } \boldsymbol{x}_i^l \text{ in } \mathcal{M}_t \\ 0 & m_i \text{ is not matched to } \boldsymbol{x}_i^l \text{ in } \mathcal{M}_t \end{cases}, \end{aligned} \tag{4.13}$$

with the indicator function $\rho(\mathcal{M}_t, m_i)$. Implementing Equation (4.13) is straightforward as it only involves counting, which we do subsequently in every algorithm cycle. The result of temporal association smoothing is the most often matched map landmark m_i^* for each detected landmark. In case the detected landmark was never matched to a map landmark, the most matched map landmark m_i^* is empty. We denote the overall set of optimal map landmark choices for all detected landmarks as

$$\mathcal{M}^* = \{m_1^*, \ldots, m_N^*\},$$

where N is the number of detected landmarks. An essential aspect of our data association approach, which includes local association, map matching, and temporal association smoothing, is that the found map associations are not based on our graph optimization results. By avoiding this feedback loop, we prevent the feedback propagation of errors and are not prone to errors like early convergence to local optima.

4.4.5 Integration of delayed measurements

Relying on a sliding window of measurements with a fixed temporal distance between poses allows us to include measurements in the state estimation even if they are delayed. While this is cumbersome with popular state estimation techniques like Kalman filters or particle filters, it is straightforward with time-based sliding window graph optimization. Our approach allows us to directly insert measurement data into the graph as long as the corresponding point in time is inside our temporal sliding window. Figure 4.18 gives an example of how we add a delayed measurement to our factor graph. In general, our approach does not have any hard requirements on the maximum delay of input data as

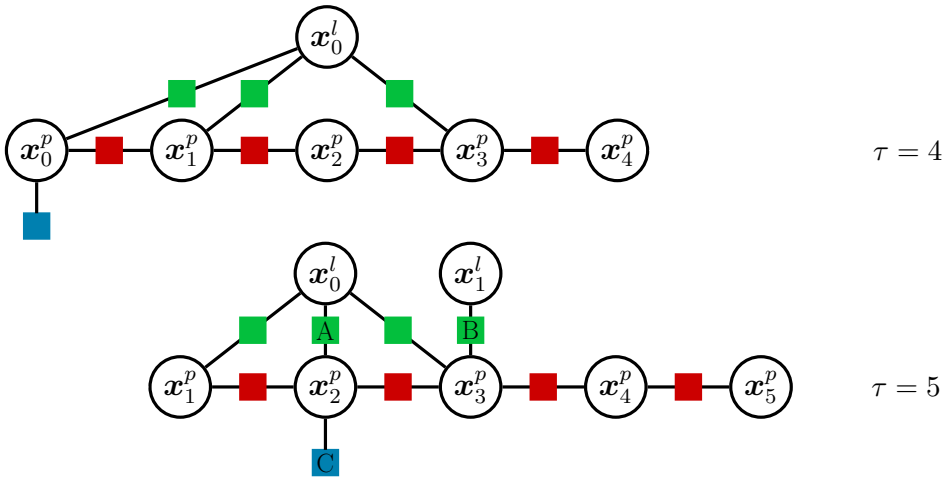


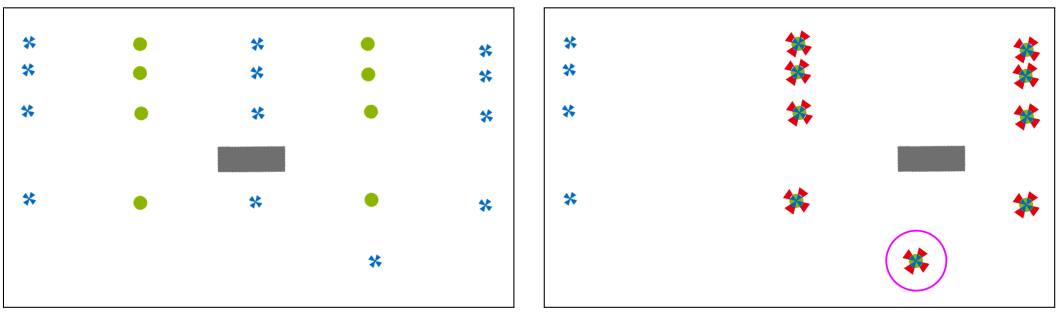
Figure 4.18: The figure illustrates three delayed measurements integrated into a sliding window graph with the maximum number of poses $N = 5$. At optimization time $\tau = 5$, the three delayed measurements A, B, and C are integrated into the sliding window graph. A is a landmark measurement of the already observed landmark $\boldsymbol{x}_0^l$, whereas B has not been observed before such that the landmark state $\boldsymbol{x}_1^l$ is created in the graph. C is a delayed GNSS measurement that constraints the pose $\boldsymbol{x}_2^p$. Including delayed measurements in our sliding window approach is a simple matter of extending the graph's structure. There is no need to alter the existing parts of the graph.

long as it stays inside the sliding window. Otherwise, the data would be lost. This property is beneficial in combination with landmark detectors that only provide detections with a low frequency (e.g., landmark detection with accumulated radar data). An exception to the required maximum delay in our approach is the odometry data. It must be available whenever a new algorithm cycle is started, and new poses are added to the sliding window. Missing odometry data leads to unconnected poses in the graph, which is problematic as it could cause optimization issues. Although we could fill the gaps with data generated by a prediction model, we aim to be model-free within our approach. Thereby, we avoid the requirement of adapting movement models to specific vehicle configurations, which makes our approach more flexible. Although integrating delayed measurements is straightforward within our approach, it is advisable to avoid measurement delays whenever possible. Adding a delayed measurement changes the underlying probability distribution, such that the optimal state estimate $\boldsymbol{x}^*$ is changed accordingly. In an extreme case where all landmark measurements are significantly delayed, the accuracy of the latest vehicle pose depends on the odometry drift's magnitude and the time span without landmark detections. In order to compensate global drift, landmarks must be matched to the given map as soon as possible. Besides our factor graph approach, the different steps of our data association architecture also allows for an effortless integration of delayed

landmark measurements. Incorporating delayed measurements in our local association and map matching is straightforward. Both steps neither require the data to arrive in order nor have hard requirements on the maximum delay. While in the literature, the integration of delayed measurement is often neglected, it is crucial for real-world applications. From our perspective, delayed measurement integration is one of the properties that make sliding window graphs superior to other commonly used state estimation techniques, especially filtering techniques.

4.4.6 Delayed associations

To avoid including possibly wrong matches, we delay association decisions until they are confident. Doing so represents a risk-averse behavior and is important in the context of automated driving. For example, we require that a landmark measurement cluster in local association needs at least a minimum amount of measurements before we consider it for map matching. This is based on the fact that the position of the local landmark cluster is closer to the expected value, the more measurements are incorporated. In return, the input data for map matching is less noisy, which improves our map matching. Similar to that, we only consider map matching results if we could at least match a minimum number of landmark clusters to the given map. This is especially important in the initialization phase when ambiguities are still unresolved. We wait to include the found map matches $\mathcal{M}^*$ until we confirmed them several times. Figure 4.19 illustrates a situation with ambiguities, whereas Figure 4.20 shows how a delayed association changes the structure of the graph.



(a) (b)

Figure 4.19: The figure shows an example situation in which delayed association is necessary. An ambiguous situation is resolved over time. (a) The vehicle pose (gray) is undetermined as it could either be on the left or the right side of the image. The correct map associations between map landmarks (blue) and observations (green) are unclear. (b) With an additional landmark detection over time (purple circle), the setting is correctly assessed by our algorithm, and all true map associations (red) are found.

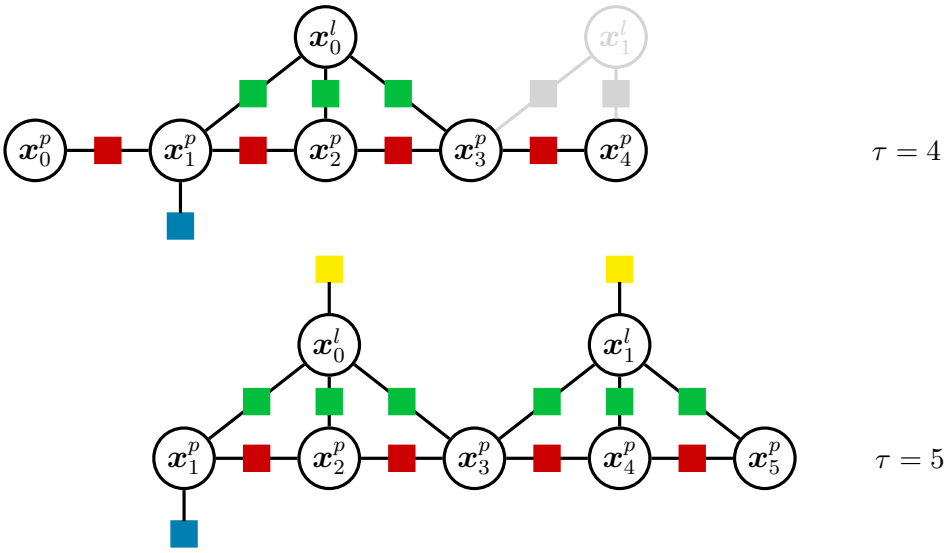


Figure 4.20: The figure illustrates two different aspects of delayed association. These are the delayed integration of landmark measurements and delayed map association. In this example, a landmark must be measured at least three times before it gets included in the graph. At time $\tau = 4$, two measurements of landmark $\boldsymbol{x}_1^l$ are available but not included in the graph. In the next algorithm cycle at time $\tau = 5$, three measurements of landmark $\boldsymbol{x}_1^l$ are available such that the landmark and its measurements are now included in the graph. Besides, both landmarks are now reliably matched to the map such that we constrain them with map factors.

4.4.7 Reversible associations

The ability to reverse falsely integrated map associations is an attractive property of our approach. Having a time-based sliding window in which we estimate all states in each algorithm cycle allows us to change the graph's structure and, thus, previous association decisions. In comparison, filtering techniques like Kalman filters marginalize out previously estimated states and their connected measurements, which introduces linearization errors that might even be critical if false map associations are marginalized. Within localization for road vehicles, the viewing angle of a landmark typically changes while driving. As an effect, the position estimate of a landmark in the local map l gets more precise. In return, our algorithm might change the associated map landmark. In practice, this manifests in neighboring landmarks that are first mistaken and then correctly associated over time. Figure 4.21 shows how a reversed association changes the graph structure, whereas Figure 4.22 illustrates a real-world situation.

4.5 Error functions

Apart from deciding which measurements are included during graph construction, it is essential how we design each factor's corresponding error function. We distinguish between the error functions for odometry, absolute pose priors, relative

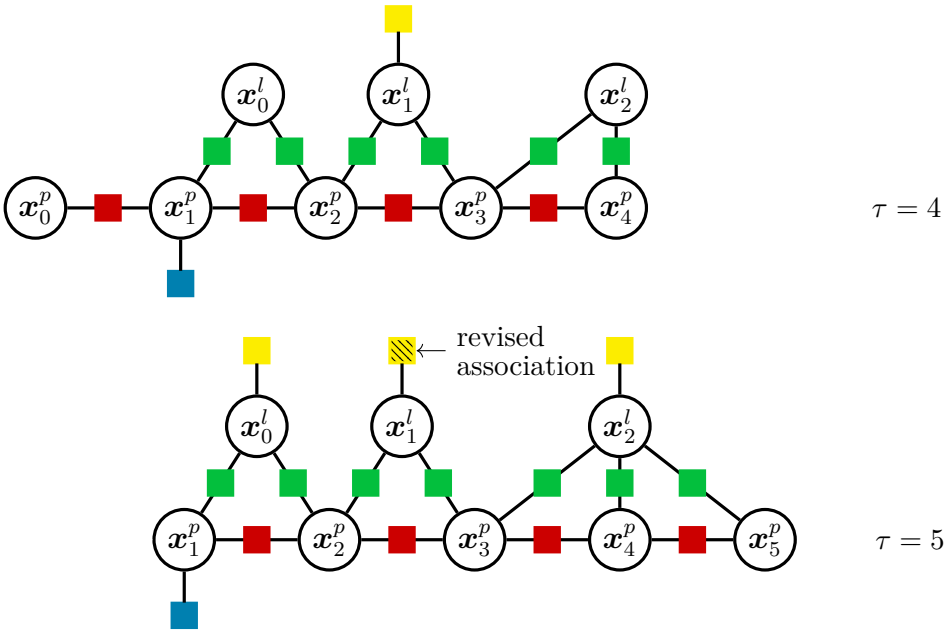


Figure 4.21: The figure illustrates an exemplary sliding window graph in which we revise a map association. New measurements, available in the algorithm cycle at time $\tau = 5$, expose that the previously made map association of landmark $\boldsymbol{x}_1^l$ at time $\tau = 4$ was incorrect. We revise the map association and add the correct map landmark to the landmark state $\boldsymbol{x}_1^l$. A revised association does not change the structure of the factor graph but instead changes the underlying constraint. To emphasize this, we illustrate the revised factor with a dashed pattern. In addition, this example shows that the two other landmarks, $\boldsymbol{x}_0^l$ and $\boldsymbol{x}_2^l$, are now associated to the map as well.

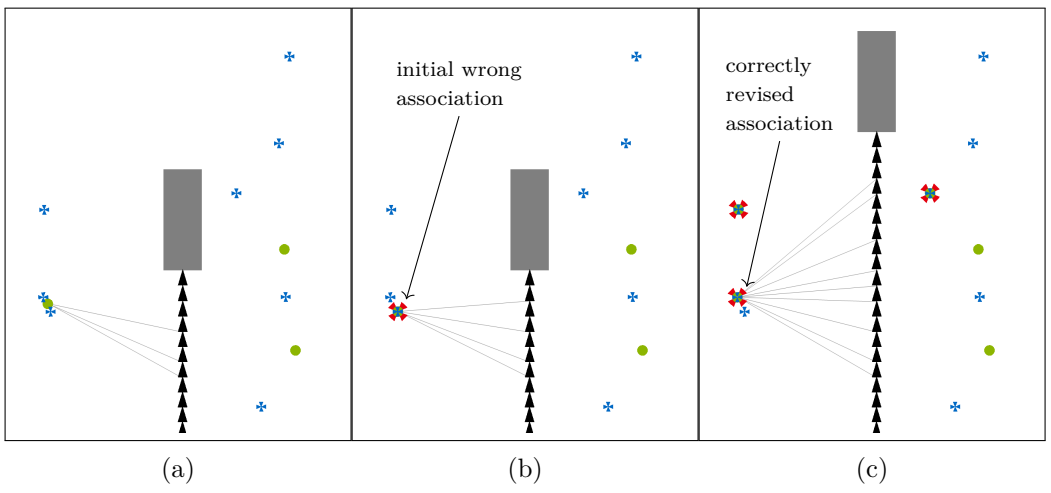


Figure 4.22: Illustration of a situation in which we revise a map association after receiving additional landmark detections. For clarity, we only show the connections (gray lines) to the revised landmark. (a) Initial unclear situation in which the landmark detection is not associated to the map. (b) An additional inaccurate landmark detection induces a false map association. (c) The situation is revised after additional landmark detections were made.

landmark, and map constraints. We construct the landmark measurement error functions in a general-purpose way such that they are not tailored to a specific sensor. In return, our approach is easily extensible and transferable. As a common general-purpose interface, we require that all landmark measurements are given in the VRF.

In the context of SLAM algorithms, error functions are often designed as the difference between a measurement $\boldsymbol{z}_i$ and a measurement function $h_i(\boldsymbol{x})$, which returns an expected measurement based on the current state estimate $\boldsymbol{x}$. The corresponding notation is

$$\mathrm{e}(\boldsymbol{z}_i, h_i(\boldsymbol{x})) = \boldsymbol{z}_i - h_i(\boldsymbol{x}).$$

See Section 3.4.1 for more details on the probabilistic interpretation.

In the following, we define the error functions used in this thesis without explicitly denoting the measurement function. Instead, we plug in the measurement function and denote the error functions directly depending on the state vector, which simplifies our notation. Our error functions for odometry measurements connect two subsequent vehicle poses and are based on the definition of Merfels (2018) and Kümmerle et al. (2011a). We integrate odometry factors as binary constraints and write the error function as

$$\mathrm{e}^{\mathrm{odo}}(\boldsymbol{x}_a^p, \boldsymbol{x}_b^p, \boldsymbol{z}_i^{\mathrm{odo}}) = \begin{bmatrix} R_{\theta_\Delta}^\top \left(\begin{bmatrix} x_\Delta \\ y_\Delta \end{bmatrix} - R_{\theta_a}^\top \left(\begin{bmatrix} x_b \\ y_b \end{bmatrix} - \begin{bmatrix} x_a \\ y_a \end{bmatrix} \right) \right) \\ \theta_\Delta - \theta_b - \theta_a \end{bmatrix},$$

with vehicle poses $\boldsymbol{x}_a^p = [x_a, y_a, \theta_a]^\top$ and $\boldsymbol{x}_b^p = [x_b, y_b, \theta_b]^\top$, odometry measurement $\boldsymbol{z}_i^{\mathrm{odo}} = [x_\Delta, y_\Delta, \theta_\Delta]^\top$, and rotation matrices R. We assume that the vehicle moves from pose a to pose b. Our error function is similar to the one used by Grisetti et al. (2010), but we arrange expected measurement and measurement the other way around. As all involved errors are squared during optimization, the order is interchangeable.

We integrate absolute pose measurements (e.g., from GNSS) as priors over individual vehicle pose states $\boldsymbol{x}_i^p$ and denote the corresponding error functions of these unary constraints as

$$\mathrm{e}^{\mathrm{abs}}(\boldsymbol{x}_i^p, \boldsymbol{z}_i^{\mathrm{abs}}) = \begin{bmatrix} R_{\theta_{\boldsymbol{z}}}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} (\boldsymbol{z}_i - \boldsymbol{x}_i^p) = \begin{pmatrix} R_{\theta_{\boldsymbol{z}}}^\top \left(\begin{bmatrix} x_{\boldsymbol{z}} \\ y_{\boldsymbol{z}} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right) \\ \theta_{\boldsymbol{z}} - \theta \end{pmatrix},$$

with measurement $\boldsymbol{z}_i^{\mathrm{abs}} = [x_{\boldsymbol{z}}, y_{\boldsymbol{z}}, \theta_{\boldsymbol{z}}]^\top$, vehicle pose $\boldsymbol{x}_i^p = [x, y, \theta]^\top$, and rotation matrix $R_{\theta_{\boldsymbol{z}}}^\top$. In cases where the measurement only constrains the vehicle position, i.e., $\boldsymbol{z}_i^{\mathrm{abs}} = [x_{\boldsymbol{z}}, y_{\boldsymbol{z}}]^\top$, we modify the error function to

$$\mathrm{e}^{\mathrm{abs}}(\boldsymbol{x}^p, \boldsymbol{z}_i^{\mathrm{abs}}) = \begin{bmatrix} x_{\boldsymbol{z}} \\ y_{\boldsymbol{z}} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix},$$

such that the error function and its Jacobian do not influence the pose orientation. In this case, it is important to ensure that the system matrix H has full rank, i.e., the vehicle orientation is well defined, to avoid any rank-deficiency problems during optimization.

4.5.1 Point-based landmark constraints

Essential to our localization approach is how we integrate landmark measurements. We assume that any detector in the vehicle outputs relative measurements in the VRF. In comparison to integrating error functions designed for a specific sensor and work in the SRF, this allows us to be sensor independent so that we can easily integrate different sensor modalities. We write the error function of a relative landmark measurement as a binary constraint that describes the relation between the vehicle pose and a single position landmark as

$$\mathrm{e}^{\mathrm{lm}}(\boldsymbol{x}_i^p, \boldsymbol{x}_i^l, \boldsymbol{z}_i^{\mathrm{lm}}) = \begin{bmatrix} x_{\boldsymbol{z}} \\ y_{\boldsymbol{z}} \end{bmatrix} - R_\theta^\top \left(\begin{bmatrix} x_l \\ y_l \end{bmatrix} - \begin{bmatrix} x_p \\ y_p \end{bmatrix} \right),$$

with vehicle pose $\boldsymbol{x}_i^p = [x_p, y_p, \theta]^\top$, landmark state $\boldsymbol{x}_i^l = [x_l, y_l]^\top$, and landmark measurement $\boldsymbol{z}_i^{\mathrm{lm}} = [x_{\boldsymbol{z}}, y_{\boldsymbol{z}}]^\top$ in VRF. This point-based error function alone is already very powerful in the context of urban automated driving as it covers a variety of landmark types. It is applicable for, e.g., pole landmarks, lamp posts, trees, and, in general, whenever the landmark state is defined as a single position.

4.5.2 Orthogonal landmark constraints

Whenever a landmark can not be fully observed by a single measurement, its state can not be estimated without further assumptions. It is necessary to either postpone the state estimation until enough measurements have been collected or break up the full state into partial estimates. Both cases are not easy to implement as they require sophisticated data management or might even not be possible, e.g., if the extent of a landmark is much larger than the sliding window. Examples of such a landmark in urban scenarios are walls of larger buildings and curbs. In rural scenarios and on highways, the predominant examples are solid lines and guard rails, which typically extend over several kilometers. Due to the vehicle's limited sensor range, they can never be measured to their full extent at once but only in parts. However, it is important that these landmarks are still considered for localization as they provide valuable information about the lateral location of the vehicle inside a lane. We assume that the corresponding measurements are available as, or can be converted to, a sequence of point measurements. Compared to *clothoid* representations, using *polylines* allows us to avoid assumptions about the shape of the measurement. Additionally, this enables us to treat

these measurements in a generic way. We can cover different landmark types with a single error function, which is beneficial for the transferability and applicability of our approach. The idea is to constrain the distance between a single support point of a measured polyline and the segment of a map polyline that has been observed. The resulting factors are unary constraints that can be compared to constraints in map-based pose graph localization. This means that there is no restriction on the number of support points in a polyline measurement as each support point is treated individually. If necessary, the polyline can be downsampled or upsampled to reflect the desired shape better. The more complex the shape and the more constraints can be generated, the better constrained is the vehicle state. Figure 4.23 illustrates the structure of our sliding window graph including multiple unary polyline factors. We follow the derivation of Bohlke (2019) and denote the error function for a single support point constraint as

$$
\begin{aligned}
\mathrm{e}^{\mathrm{poly}}(\boldsymbol{x}_i^p, \boldsymbol{z}_i, \boldsymbol{m}_1, \boldsymbol{m}_2) &= \frac{\det\left(\begin{bmatrix}\boldsymbol{u} & \boldsymbol{v}\end{bmatrix}_{2\times 2}\right)}{|\boldsymbol{v}|} \\
\boldsymbol{v} &= \boldsymbol{m}_2 - \boldsymbol{m}_1 \\
\boldsymbol{u} &= \hat{\boldsymbol{z}} - \boldsymbol{m}_1 \\
\hat{\boldsymbol{z}} &= R_\theta \boldsymbol{z}_i + \begin{bmatrix} x_p \\ y_p \end{bmatrix},
\end{aligned}
\tag{4.14}
$$

with vehicle pose $\boldsymbol{x}_i^p = [x_p, y_p, \theta]^\top$, globally transformed measurement $\hat{\boldsymbol{z}}$, support points $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ of the map line segment, $|\boldsymbol{v}|$ is the norm of vector $\boldsymbol{v}$, and $\boldsymbol{z}_i \in \mathbb{R}^{2\times 1}$. A disadvantage of integrating polyline landmarks in such a generic way is that estimating possible map updates can not be done within graph optimization. Without further assumptions, the support points $\boldsymbol{m}_1, \boldsymbol{m}_2$ are underconstrained as they can move freely along the described line without changing the error function. As a consequence, we do not estimate the landmark position and treat the error function as a unary pose constraint. Any map updates regarding polyline landmarks can be done in a separate module after graph optimization, or further assumptions must be made.

Our error function assumes that the association between the measurement support point and the map segment has already been found upfront. As we treat each support point of a measurement individually, each measurement can directly be included in the graph without local association. However, finding the segment of a polyline landmark requires an adjusted map matching, which is explained by Bohlke (2019) in more detail.

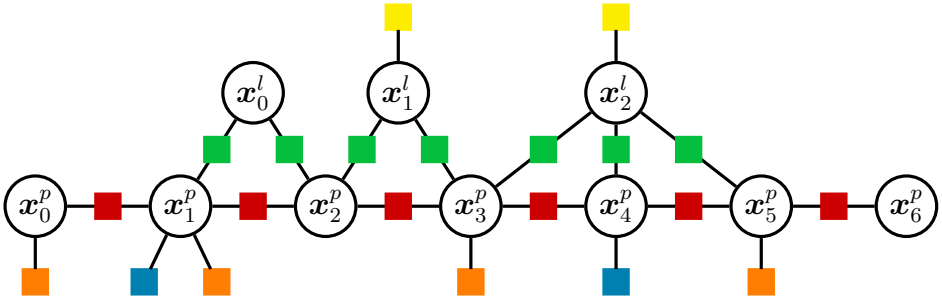
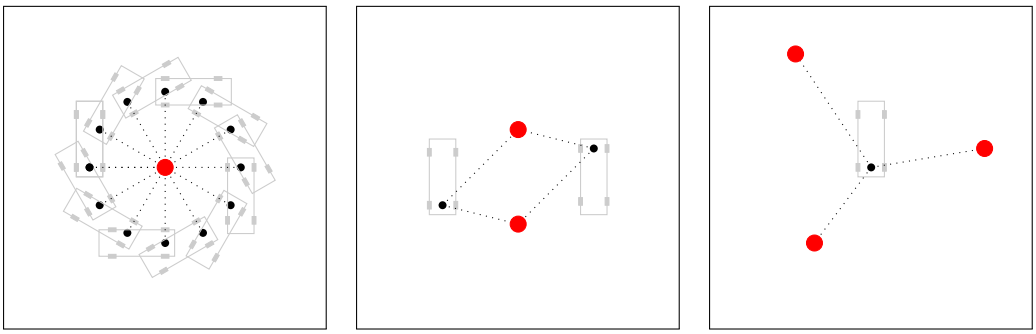


Figure 4.23: The figure illustrates the structure of a sliding window graph with unary polyline factors, shown in orange. In comparison to absolute pose constraints (in blue), unary pose-landmark factors only constrain the vehicle pose along one direction. Note that the factor graph here only allows inferring which parts of the system matrix H are affected by the unary constraints. The actual error function is given in Equation (4.14).

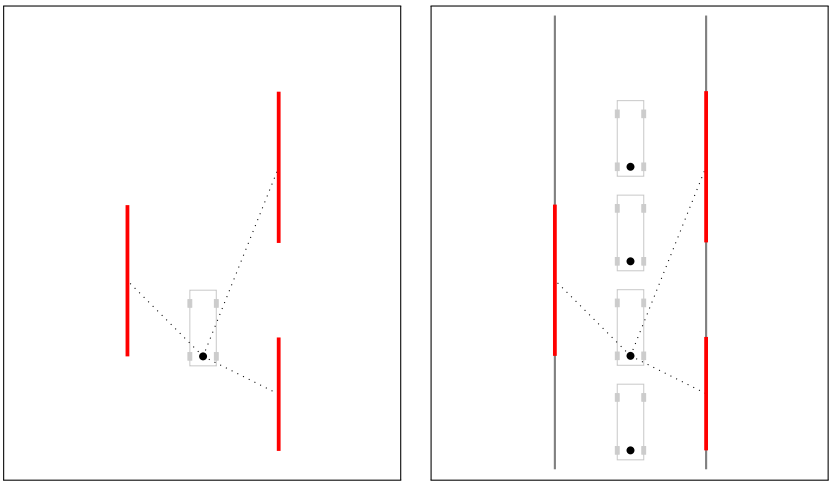
4.5.3 Fully constraining the sliding window graph

A vital aspect of sliding window graph localization is ensuring that the underlying optimization problem has a unique solution. Therefore, the system matrix H must be fully constrained, i.e., it must not suffer from rank-deficiency. A foundation of our graph-based localization approach is that it relies on globally constraining the position of landmarks based on map priors, which implicitly globally constraints the vehicle poses. While landmark and odometry measurements only provide relative information, the map priors contribute to finding unique global pose estimates. Similarly, GNSS may also globally constrain the sliding window graph. Due to possible outages, this is, however, not a reliable option. Therefore, we add an artificial pose constraint based on previous pose estimates to the end of the sliding window graph in cases where the graph would be globally underconstrained. Other options to fully constrain the optimization problem are using Levenberg-Marquardt optimization, which inherently ensures optimizability through regularization. Compared to Gauss-Newton optimization, Levenberg-Marquardt tends to be slower, which is why we prefer artificially altering the optimization problem by adding an additional constraint. Ensuring that the sliding window graph is fully constrained is, in practice, easier with point-based landmark constraints than with orthogonal landmark constraints. This is based on the spatial distribution of underlying landmarks for which the error functions are designed. Figure 4.24 illustrates how point-based landmark constrain a vehicle pose, whereas Figure 4.25 demonstrates the ambiguity challenge typical for orthogonal constraints based on road marking measurements.



(a) (b) (c)

Figure 4.24: The figure illustrates that at least three point-based landmark measurements are required to fully constrain a single vehicle pose. In our graph-based sliding window approach, we optimize over a set of past measurements such that the landmark measurements can be made at different points in time to resolve this kind of ambiguity. Note that we here only illustrate constraining the vehicle pose in some local coordinate system. Globally constraining the vehicle within a map, however, is similar. (a) In the case of a single landmark measurement, the vehicle could be anywhere on a circle around the measured landmark. (b) Assuming that the map association is unclear, there are two options for the vehicle's location. (c) Three landmarks fully constrain the vehicle pose such that the vehicle's location is distinct.



(a) landmark measurements (b) ambiguities with unary constraints

Figure 4.25: The figure illustrates an underconstrained situation with orthogonal constraints based on road markings. Our graph-based approach resolves these types of ambiguities by incorporating past measurement data from other landmarks or GNSS data in the sliding window, which is not illustrated here. (a) Exemplary polyline landmark measurements of road markings in the vehicle reference frame. (b) The detected landmark measurements only allow constraining the vehicle laterally to the road markings. Without further information, the vehicle can not be localized longitudinally. In addition to the illustration, the vehicle could also be rotated by 180°, which would also satisfy the constraints.

4.6 Time synchronization

Our choice to use a fixed pose frequency in each sliding window requires synchronizing measurement data to the timestamps of the poses in the graph. In our approach, we perform the synchronization right before we add the measurements to the graph. The prior data association steps are not affected as they are computed independently from the graph's pose frequency. We distinguish between three main synchronization principles that are suitable depending on the type of measurement data. We explain the three types in the rest of this section. Figure 4.26 illustrates the time synchronization issue.

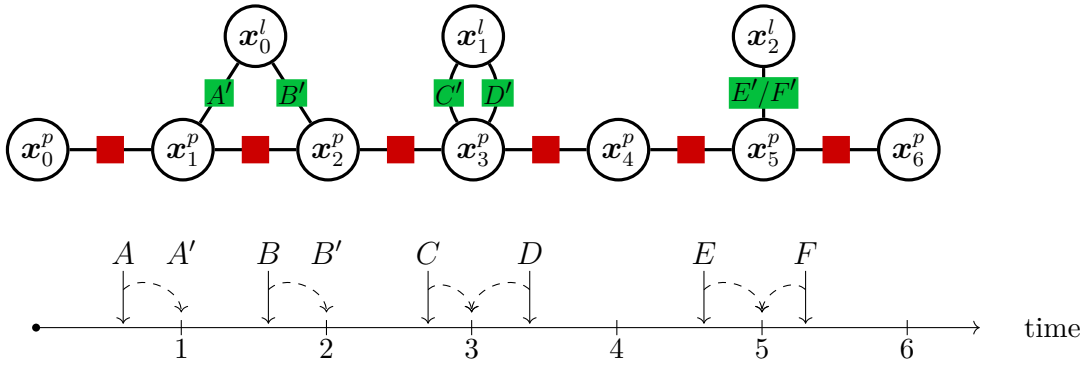


Figure 4.26: The figure illustrates the necessary time synchronization for measurement data to be included in the sliding window graph. The landmark measurements A, B, C, D, E, and F, are measured at some timestamp that does not match the pose timestamps in the sliding window graph. Therefore, we project each measurement to the nearest timestamp of a vehicle pose. We mark the projected measurements with the prime symbol ($'$). For clarity, we omit to mark the projections of C, D, E, and F. Furthermore, the figure illustrates two options for handling multiple measurements between the same pose and landmark. In the case C' and D', we add two individual constraints to the graph, whereas in the case E' and F' we combine both measurements by averaging them.

The most straightforward option is to ignore the time difference between measurement and graph pose. In this case, we overwrite the measurement's timestamp with the timestamp of the temporally closest graph pose. At the same time, we do not change the rest of the measurement, e.g., the part of the measurement given in the VRF or WRF. This is a suitable choice whenever the pose frequency is high enough such that the induced error is marginal. It can be a valid option for GNSS and landmark measurements.

The second option is to linearly interpolate between consecutive measurements. While this is suitable for absolute pose measurements because they are taken in a WRF, it is unsuitable for landmark measurements. Since landmark measurements are taken in the VRF, measurements of the same object change if the vehicle moves, which renders interpolation impractical. For absolute measurements, it is important that interpolation easily correlates the data whenever

an original measurement is used twice. Figure 4.27 illustrates this issue. Within graph optimization, this type of error leads to overconfident estimates. One way of avoiding this is to always consume two measurements for computing one interpolated measurement. As this halves the measurement frequency, it must be considered in practice if it is suitable for the actual implementation. Another approach is to consider the induced correlation within the optimization as, e.g., presented by Merfels et al. (2016).

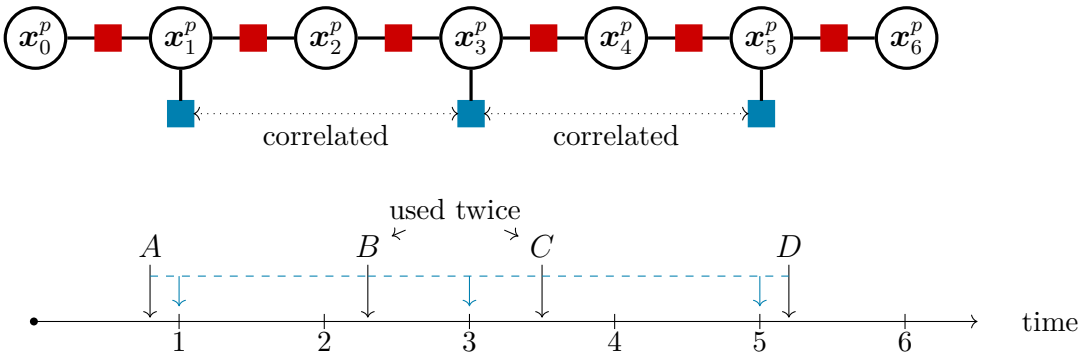


Figure 4.27: Interpolated and thus correlated measurements arise whenever a measurement is used twice for interpolation. The figure shows the original absolute pose measurements A, B, C, and D, which are used for creating the interpolated blue factors. The measurements B and C are used twice, which introduces a correlation between the interpolated consecutive factors in the sliding window graph.

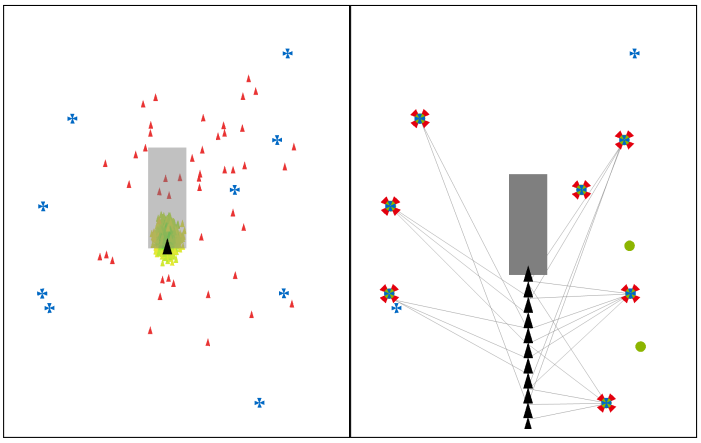
The third option is to project measurements in time with the help of odometry data. For absolute measurements, this is possible if the measurement also contains the vehicle's orientation. Depending on the time frame that is interpolated and the heading's accuracy, this option might amplify the measurement errors, which is why this option should be avoided in practice. On the contrary, for landmark measurements, this option is perfectly valid. We use the odometry to compensate for the vehicle movement and change the landmark measurement accordingly. Although this induces correlation between the odometry and landmark measurements, the effect is negligible if the odometry has little drift and the interpolated time span is short as it is commonly the case in the context of automated driving. We require that the odometry's measurement frequency is higher than the pose frequency, which allows us always to downsample the odometry as required for projection and reduce interpolation errors.

Choosing between the discussed synchronization methods mainly depends on the desired graph's pose frequency and how distinct the discussed advantages and disadvantages are. In general, the higher the pose frequency, the less synchronization errors. In our case, the odometry is available with 100 Hz, GNSS measurements are available at 1 Hz, landmark measurements typically arrive with a frequency of 10 Hz, and the frequency of poses inside the graph typically is

$f = 25\,\text{Hz}$. Thus, the maximum time span for synchronization that we need to cover is 40 ms. We choose to interpolate GNSS measurements and ignore the induced correlation, whereas we project landmark measurements with odometry measurements.

4.7 Particle filter vs. sliding window graphs

In the following, we compare our graph-based sliding window localization to particle filters on an argumentative level. At the end of this section, Table 4.2 provides a brief overview of our assessment results. We compare the stochastic nature of particle filters against the deterministic optimization of graph-based approaches, discuss multimodal vs. unimodal pose distributions for automated driving, and characterize conceptual advantages of graph-based approaches over particle filters. In detail, these are the capability of estimating old poses and the integration of outdated measurements. We provide a visual comparison of both concepts in Figure 4.28.



(a) Particle filter (b) Graph-based localization

Figure 4.28: Localization problem in the same situation solved with two different approaches. (a) Particle filter. The color of a particle represents its current weight. If a particle weight is high the color ranges from green to yellow. Recovery particles from GNSS receive a low weight (red). (b) Sliding window graph. Finding matches (red crosses) between landmark observations (green dots) and map landmarks (blue crosses) allows us to constrain the vehicle trajectory (black triangles). The connections (gray lines) illustrate from which poses a landmark was observed. The current vehicle pose is depicted as a gray rectangle.

Stochastic vs. deterministic algorithm

A fundamental difference between particle filters and graph-based techniques is how they update their state belief. Graph-based approaches use optimization algorithms like Gauss-Newton to find the optimal state that best fits the current

set of measurements. In practice, the optimization starts at some initial state estimate and iteratively improves its estimate (see Section 3.4.2). This process can be considered deterministic as the same input data, i.e., measurements and initial state, always produces the exact same state vector output. In comparison, the resampling step in the particle filter algorithm requires drawing samples from a distribution, which is why particle filters are stochastic. As a consequence, particle filters produce different results on multiple runs with the same input data. Although the variations in the results might only be marginal, the implications w.r.t. functional safety are unclear. Due to the safety-critical role of localization in automated driving, it might be that stochastic algorithms, like particle filters, are inapplicable for the series development of automated driving functions. Although this remains under discussion, deterministic localization approaches are favorable, in our opinion.

Multimodal vs. unimodal state estimation

Graph-based approaches and particle filters follow different concepts in representing the probability distribution of the vehicle's pose. In general, particle filters are able to represent multimodal beliefs about the vehicle's pose. An advantage of particle filters is that it is straightforward to incorporate environmental ambiguities into the nonparametric particle distribution. In graph-based localization under the conventional probabilistic interpretation, the graph infers the set of most likely poses given the measurements, see Equation (3.1). As a graph is usually built up by multiple poses, this is technically a multimodal Gaussian distribution. Each pose, however, corresponds to a single unimodal Gaussian that is being estimated. Consequently, any environmental ambiguity must be resolved during graph construction. Therefore, the expressiveness about the belief of the pose at a given point in time is higher for particle filters, as they are not restricted to parametric or Gaussian distributions. Depending on the use case and subsequent software modules for which localization is required, both approaches can be feasible. An issue with particle filters for localization is that it is necessary to identify a specific particle as the pose estimate is sent out to other subsequent modules. Since particle filters are able to represent multiple modes, the chosen pose estimate might jump between different equally probable modes of the distribution. Although some heuristics, as, e.g., combining multiple particles into a single estimate, may partly relax the issue, the underlying problem remains. For example, taking the mean over all particles might ignore that the underlying probability distribution has multiple modes such that the mean estimate could lie within a near-zero probability region between modes. With respect to automated driving, we prefer using unimodal estimates for localization if the subsequent modules require one pose estimate per timestamp.

Integration of outdated measurements

Incorporating outdated and out-of-sequence measurements into the pose estimation is a beneficial property for real-world localization applications. An advantage of sliding window graphs is that the pose history maintained in the graph makes it easy to integrate delayed or out-of-sequence measurements. Similarly, the sliding window graph's flexible structure makes it possible to perform delayed data association so that only confirmed map matches are incorporated in the graph. An additional benefit is that the mechanism of maintaining a history of poses and measurements allows the continuous relinearization of all states within the sliding window such that the overall linearization errors are reduced. In contrast, particle filters usually only track the most recent vehicle pose within each particle's state. The set of particles is updated based on propagating previous pose estimates and incorporating only the latest measurements. Incorporating out-of-sequence or delayed measurements with particle filters would require rewinding the state of the particle filter to a specific point in time, adding the delayed measurement, and recomputing all following steps. Alternatively, all delayed measurements must be projected to the latest timestamp, which potentially introduces great errors depending on the time span of the projection. In principle, methods like maintaining a history of particle sets or calculating back in time may enable incorporating delayed measurements within particle filters. However, in practice, delayed or out-of-sequence measurements are usually discarded because of the required computational resources or memory. The same arguments hold for delayed landmark association, which is unfeasible for particle filters. Another challenge with particle filters is that the measurements incorporated in each algorithm cycle must share the same timestamp as the vehicle pose that is being computed, which introduces interpolation errors. Although methods like keeping particles for various timestamps could help to reduce the incorporated interpolation errors, the disadvantages of tracking more particles outweigh the advantages. In comparison, our sliding window graph's advantage is that the measurements must not be synchronized to the latest vehicle pose. However, each measurement must be individually interpolated to its temporally closest pose maintained within the sliding window graph. As long as the temporal distance between subsequent poses in the graph is smaller than the particle filters output interval, the introduced interpolation errors within sliding window graphs are smaller than the interpolation errors within particle filters. In theory, sliding window graphs could also contain poses for each measurement timestamp, which could potentially eliminate interpolation errors. This contradicts our approach of having a fixed temporal distance between poses, which heavily reduces the number of required poses in the graph and thus improves the time required for data management and optimization. Overall, sliding window graphs provide

much higher flexibility in including time-based measurement data than particle filters.

Estimating past poses

Graph-based approaches maintain a recent history of poses within the state estimation problem, while particle filters only maintain the current pose. This allows graph-based approaches to benefit from delayed data association and frequent relinearization, which means that the past and present states are simultaneously improved when, e.g., past associations are revised. Additionally, this allows us to output a *lagged pose*, i.e., a pose with a certain delay compared to the current timestamp, representing the current best estimate of the past. Outputting a lagged pose can be beneficial for applications that need older pose estimates in favor of higher estimation accuracy. In comparison, particle filters never change their past beliefs about the robot poses. They only track the current set of particles and thus lack the option of outputting a lagged pose.

	particle filter	graph-based
algorithm approach	stochastic	deterministic ✓
solution space	multimodal	unimodal ✓
integrating outdated measurements	✗	✓
estimating old poses	✗	✓

Table 4.2: Summary of our argumentative comparison between particle filters (PF) and graph-based localization (GBL) for automated driving. Checkmarks (✓) indicate that the related characteristic is favorable for our automated driving application. A cross (✗) denotes an unsupported property.

4.8 Summary

In this chapter, we presented our graph-based sliding window approach for vehicle self-localization. We first discussed our design principles, including that our data association does not rely on estimated vehicle poses of previous algorithm cycles to avoid accumulating errors. Additionally, we require that landmark measurements are given in the VRF, which allows us to integrate landmarks from different sensors in a generic way. We presented our graph-based sliding window approach that relies on a fixed pose frequency, is fast enough for application in an automated vehicle, and integrates odometry, landmark measurements, and GNSS. The latter is only required for global initialization or as a fallback in unmapped areas. Our presented data association approach is to, first, locally cluster landmark measurements that belong to the same landmark. Afterward, we perform a subsequent map matching step that assigns the identified landmark clusters to

a given general-purpose third-party landmark map. Finally, we aggregate the found map matches of each algorithm cycle such that we can extract temporally smooth associations and revise and delay associations if necessary. Our data association concept copes with the challenge of using a given general-purpose third-party map for localization. We proposed and discussed our approach that allows us to find the best overlay between the landmarks stored in the given map and the landmarks detected by the vehicle at runtime. Furthermore, we discussed including different error functions for different landmark types, elaborated on synchronizing measurement data for including it in our sliding window graph, and finally compared our approach to particle filters for localization. In sum, our presented graph-based sliding window approach contributes to vehicle localization in the context of automated driving.

Chapter 5

Delayed map refinements

One of the main challenges of localization with given landmark maps are environmental changes that render the used landmark maps outdated over time. In the following, we extend our graph-based localization approach for automated driving w.r.t. selectively updating and extending given maps. We introduce our concept of keeping landmark maps up to date through computing map updates live in the vehicle and validating them through a back-end server before actually changing the map. In short, we call this *delayed map refinements*. We separate the map update process into the independent parts deleting, modifying, and adding landmarks. Our contribution and focus is the estimation of landmark positions within our sliding window approach, whereas the server-side validation is not part of this thesis. We introduce *sparse global priors* that capture the information of measurement data that we remove from the sliding window by approximating marginalization. This chapter is in parts closely related to the Master's thesis of Rumberg (2018), which I co-supervised, and based on our previously reported findings (Wilbers et al., 2019c). We repeat, extend, and discuss our results in this chapter.

5.1 Refining maps for automated driving

Construction works and environmental changes influence any localization system that is based on given maps. Typical examples in urban areas are road construction works and the demolishing and construction of buildings. As a consequence, the given maps become outdated over time, which may lead to localization failure. Therefore, it is crucial to update maps over time. Since inaccurate and erroneous map updates may lead to inaccurate localization and, therefore, hazardous driving behavior, updating maps is a safety-critical task. For this reason, we consider it essential that map updates are verified before they are applied. We assume that this is achieved by aggregating fleet data in a back-end service,

which computes and distributes reliable map updates over-the-air. An essential aspect of validating updates through a back-end server that we need to consider when designing an update process is how we generate the input made available to the back-end. We discuss this acquisition process in the following. Nowadays, maps for automated driving are mostly created and updated based on particular mapping drives. However, the trend is towards using fleet data that can be used in an automated fashion to generate map updates. Ideally, the size of the data transmitted to the back-end server is small such that the required bandwidth and resulting costs are minimal. We achieve this by transmitting preprocessed data instead of raw sensor data or raw landmark measurements. As discussed before, we have designed our graph-based sliding window approach to estimate the global position of landmarks. This has the advantage that several landmark measurements are already aggregated into a single estimate, which reduces the amount of data. Figure 5.1 illustrates our proposed update architecture. In the scope of this thesis, we focus on computing map refinements, which are position updates for existing landmarks and position estimates for new landmarks. We aim to compute these refinements whenever the vehicle is successfully localized, which enables us to compute globally accurate landmark positions. With respect to automated driving, we assume that the vehicle's navigation avoids unmapped areas. Likewise, we assume that construction zones are made available to the vehicle, e.g., comparable to traffic news, such that areas that potentially compromise localizability are avoided. These areas might be updated with the help of manual mapping drives. The extend of how sensible the localization system is to changes in the environment depends on the diversity of the incorporated landmark types and their amount. Whenever only a small part of the landmarks are outdated, our localization approach likely remains unaffected, which is a crucial property. This is based on our design choices for incorporating third-party maps, where we consider that detected and map landmarks may only partially match. The property is likewise beneficial for scenarios in which map landmarks are partially outdated or missing. We present our refinement architecture in more detail in the following section.

5.2 Adding, modifying, and deleting landmarks

Within refining landmark maps, we distinguish between the three distinct operations: adding, modifying, and deleting. While our approach relies on verifying updates through a back-end service, which itself is not part of this thesis, we outline the most important ideas in the following. An essential aspect of updating landmark maps to consider is that the initial maps might get tailored towards a specific sensor setup over time. This might occur whenever the fleet data used in

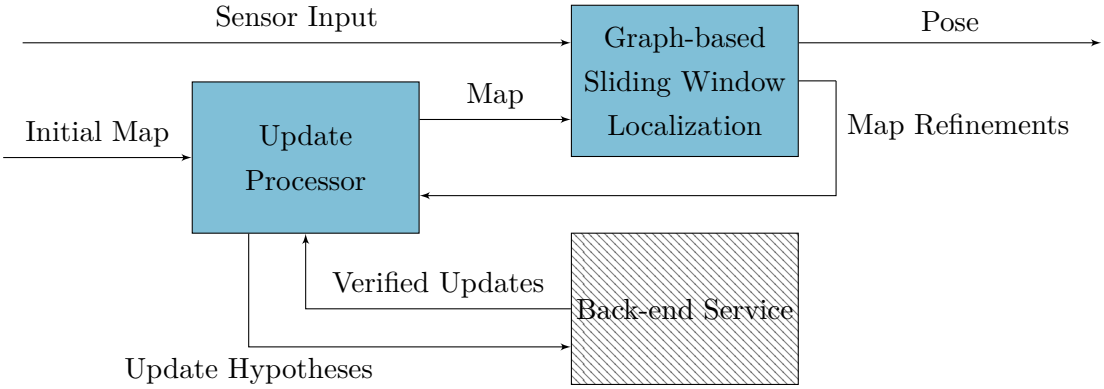


Figure 5.1: Brief overview of our map refinement architecture. We illustrate modules inside the vehicle as blue boxes and the back-end service as grayed out. In this thesis, we focus on the in-vehicle modules and assume the back-end service to be given. Our graph-based sliding window approach computes map refinements, which we collect in the Update Processor. The latter is responsible for the data exchange with the back-end server and is not further discussed in this thesis.

the back-end service is based only on a specific sensor setup. If the map contains elements that can not be detected with a specific sensor setup, the corresponding landmarks will get deleted over time. Likewise, new landmarks will only be added based on the specific sensor setup. This might affect the initial general-purpose map, which was usable with different sensors, such that it loses this property over time. Nevertheless, this can also be seen as an advantage that allows omitting expendable map elements. It might even be possible to extract and maintain different map variants for different sensor setups in the back-end service.

Adding landmarks to the map is required whenever they are missing in the initial map or were newly installed, e.g., due to construction works. We estimate the position of these landmark additions within our graph-based sliding window approach whenever our sliding window graph is sufficiently constraint by other map landmarks. A challenge in adding landmarks to the map is that the estimated landmarks might be based on false positive detections. For example, if a dynamic object, which is temporally not moving, is mistaken as a stationary landmark. Since we treat all map refinements as hypotheses for updates, we assume that the back-end server takes care of this challenge. The most straightforward solution, in this case, is that the back-end server requires a minimum amount of hypotheses by various vehicles before it confirms the update. Similarly, the back-end service is responsible for ensuring that verified additions are accurate. A simple approach here would be to average over the received hypotheses to get a verified landmark position.

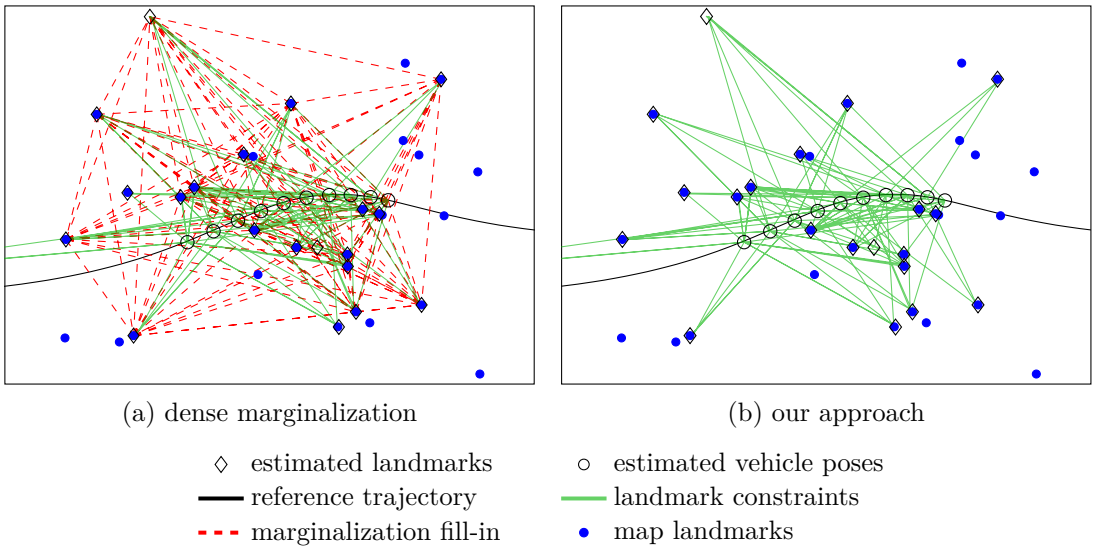
We consider modifying the position of map landmarks relevant for improving

the position accuracy within the initial map. This is especially helpful if the accuracy of the initial map is partially or entirely unknown. Our approach is to recompute the map landmark positions while using them for localization within our sliding window graph. As presented in Chapter 4.3.2, we integrate map landmarks as priors such that the position estimate within the graph may differ from the map position. These position refinements are sent to the back-end server, which compares the refinements with other fleet data before the position modifications are sent out as a verified update.

Within our update architecture, deleting landmarks is within the responsibility of the back-end service. Our approach is to aggregate a list of map landmarks that have successfully been used for localization at runtime. This list of confirmed map landmarks is sent in fixed intervals to the back-end server, which receives various fleet confirmations over time. If a landmark is never confirmed during a specific time span, i.e., never used for localization, it can be deleted from the map. This reflects a temporal decay, which means that, over time, unconfirmed landmarks will be deleted from the map. An advantage of this approach is that it does not require additional implementation as it reuses the found map matches during localization. However, a disadvantage is that confirming every map matched landmark might result in an excessive amount of data that needs to be transferred to the back-end service. This could be solved by introducing confirmation requests for selected landmarks that are sent out by the back-end service to the fleet. The vehicle would then only transmit the confirmation of the requested landmarks. An alternative approach for deleting landmarks would be to asses within the vehicle if a landmark should have been detected. This might be inferred by considering if the line-of-sight to a landmark was unobstructed and the sensors should have measured a landmark. If this is the case, the vehicle sends a deletion request to the back-end service, which is responsible for verifying the request based on fleet data. Nevertheless, if this alternative approach is feasible in practice depends on the sensor modalities used for localization. For example, it is easier to infer if the line-of-sight is blocked with a LiDAR than with camera or radar.

5.3 Estimating landmark positions

We focus on the problem of estimating the positions of previously unmapped landmarks on-the-fly in the vehicle. In particular, we use our sliding window formulation to keep the problem size computationally tractable. We present a novel approach, which approximates marginalization without inducing any fill-in, as shown in Figure 5.2. Compared to nonlinear factor recovery (Mazuran et al., 2016), generic linear constraints (Carlevaris-Bianco et al., 2014), and the pose



(a) dense marginalization (b) our approach (c)

Figure 5.2: The influence of marginalization on the graph structure. The illustration shows that our approach does not suffer from fill-in, which reduces computational complexity and thus is faster to compute. Simultaneously, our approach provides a comparable accuracy of the resulting estimate.

graph compression approach by Kretzschmar and Stachniss (2012), we focus on the special case of sliding window graphs for which different assumptions hold. On the one side, we target a special approximation topology with individual priors for each state. On the other side, we ensure by design that our system matrix H always has full rank, i.e., we do not need to cope with the deficient rank problem. Our main contribution is a novel sparsification scheme for marginalization in sliding window graphs. We achieve this by deriving individual global priors for all states involved in marginalization. This allows us to estimate incremental map refinements in the context of automated driving. In sum, we make four key claims that our approach

(i) approximates sliding window marginalization with sparse global priors,

(ii) utilizes global linearization points and thus does not require local optimization,

(iii) computes conservative landmark positions for incremental map refinement,

(iv) has exactly the same sparsity pattern compared to using no marginalization but provides more accurate estimates.

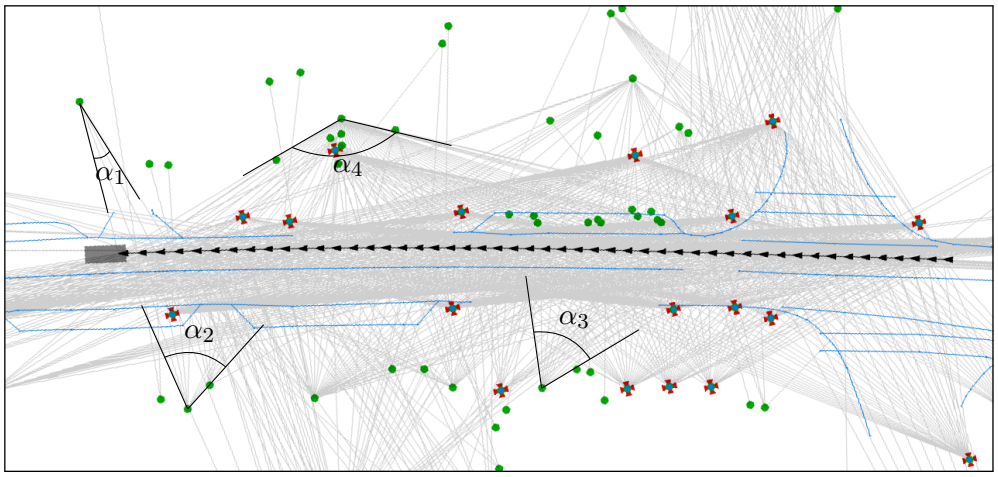
5.3.1 Selecting suitable landmarks

Before we describe our approach's mathematical details in the later sections, we first focus on the important practical aspect of filtering out unsuitable point land-

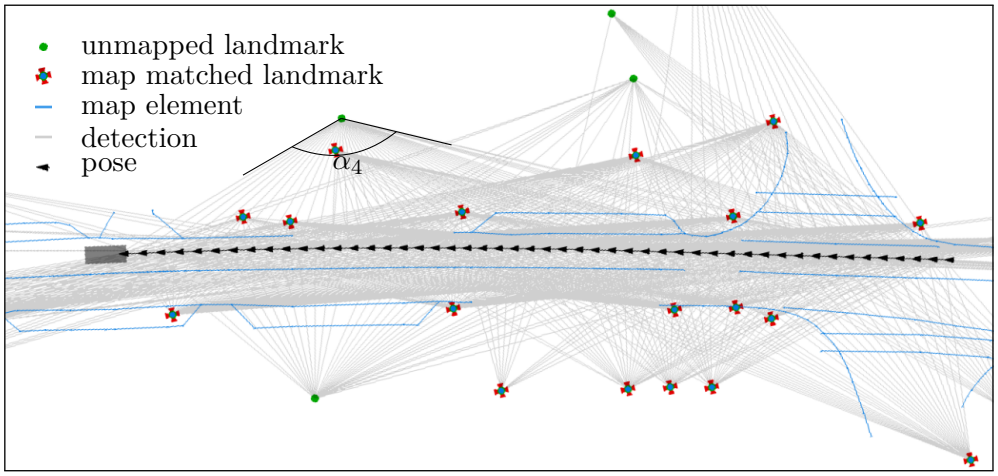
marks. Unsuitable in this context refers to false positive measurements, as well as landmarks that can only be observed by the vehicle from very distinct locations. We favor landmarks that the vehicle observes from multiple locations, which increases their impact on localization compared to landmarks that are rarely observed. The probably most straightforward filtering approach is requiring at least a minimum number of measurements per landmark before including it in the state estimation. However, the number of received measurements per landmark depends on the driving speed and might grow arbitrary, e.g., when standing still, and thus is insufficient for our intended filtering. Instead, we additionally estimate the spatial coverage of the viewing angles from which a landmark is observed. Within our approach, this is an extension that only affects the detected landmarks that we could not match to the map, which means that we suggest them as potential map additions to the back-end server. We only include unmatched landmarks in our sliding window graph if the landmark has a minimum amount of measurements and if the spatial viewing scope exceeds a prespecified threshold. To estimate the covered viewing scope of a landmark, we extend the local association step of our approach (Algorithm 2). For each landmark measurement that we add to a cluster, we project it into the local coordinate system that is based on accumulated odometry data and check how the measurements influence the currently stored viewing scope. In short, we store the two angles that define the outer boundaries of the viewing scope and extend them if necessary based on the current measurement that we integrate. We perform this iteratively for each incoming landmark measurement. Since our calculations are done in the local coordinate system, the calculated viewing scopes are affected by the accumulated drift error of the odometry. Therefore, our estimated viewing scopes are only approximations, which are more accurate the better the odometry. We illustrate the effect based on a real-world example in Figure 5.3, where we also highlight the viewing scopes of some landmarks.

5.3.2 Sliding window marginalization

In this section, we describe how to preserve the information of measurements outside of the sliding window through marginalization. A common approach for marginalizing out states and their connected measurements is to use the Schur complement, as we have shown in Chapter 3.6. Sibley et al. (2010) demonstrate this for sliding window filters and describe how the Schur complement induces fill-in in the system matrix H. Another negative aspect of marginalization is that it induces linearization errors. So far, we prevented fill-in and accumulated linearization errors by simply truncating measurements and states instead of marginalizing them out. This is a valid option for pure localization applications as long as the sliding window contains enough measurements that sufficiently con-



(a) unfiltered



(b) filtered based on viewing scopes

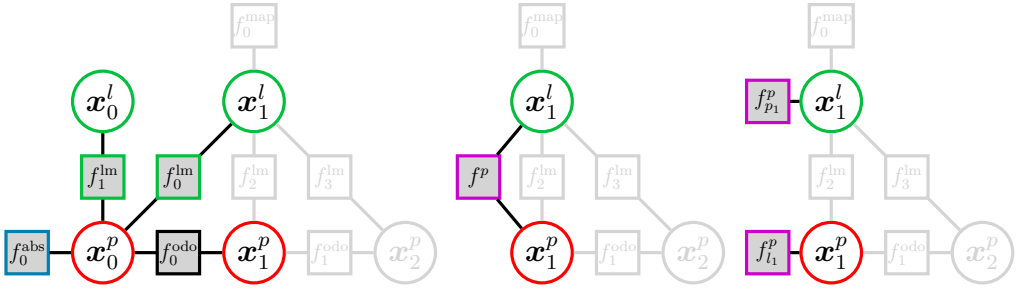
Figure 5.3: The figure shows a real-world example for filtering out unsuitable unmapped landmarks based on their viewing scope. The remaining unmapped landmarks with a suitable viewing scope are map refinements that are sent to the back-end server to extend the map. (a) The figure shows the sliding window graph without filtering out any landmarks. It also exemplarily shows the viewing scope of four different unmapped landmarks. We incorporate all landmarks in the graph that are part of local association. (a) Sliding window graph that only incorporates unmapped landmarks with a viewing scope above a predefined threshold. From the exemplarily selected landmarks, we only integrate the one with the viewing scope α_4 in the graph. The landmarks with view scopes α_1 and α_2 from the unfiltered example might get included at a later point in time as the vehicle moves forward and the viewing scopes are potentially increased. All unmapped landmarks, including those not integrated into the graph, remain part of our algorithmic approach for local association and map matching. Selecting landmarks based on their viewing scope only affects the unmapped landmarks that we estimate as potential map refinements inside our sliding window graph.

strain the graph. Whenever the graph is not well constrained by the included measurements, as discussed in Section 4.5.3, the optimization might fail, or the pose accuracy might drastically decrease, leading to localization failure. A simple heuristic that helps alleviate the issue is to condition the graph on the oldest pose in the sliding window. This effectively sets the oldest pose as an anchor of the graph such that the optimization space is restricted, which is error-prone due to noise in the pose that is used for conditioning. Besides these challenges, we will show in our evaluation that truncation is a valid option for pure localization in urban scenarios. Nevertheless, a significant drawback of truncation is that the state estimates of individual landmarks become unstable over time. Compared to the vehicle trajectory, which is constrained by all graph measurements, an individual landmark typically has much fewer constraints. As the sliding window moves past a landmark, the number of measurements inside the sliding window decreases until the landmark is no longer part of the graph. As an effect, the landmark's estimated state in each algorithm cycle depends on a different set of measurements. To prevent information loss and estimate reliable landmark positions, we investigate marginalizing out the measurements instead of truncating them. To avoid the fill-in problem, we present an approach that approximates dense marginalization with sparse priors. A difference to many graph-based approaches is that our temporal sliding window guarantees by design that there are always global factors in the marginalization blanket. We ensure this by including global information from previous time steps. Because of that, our sparsification scheme does not require relative formulations as, e.g., the approaches by Mazuran et al. (2016) and Eckenhoff et al. (2016). Thus, we avoid any deficient rank issue in cases where we only marginalize measurements in the VRF.

5.3.3 Calculating the marginalization prior

Our sliding window approach is based on marginalizing out the oldest pose and adding a new pose to the front of the graph in every timestep. As a consequence, the number of poses in the graph is constant, whereas the environment naturally bounds the number of landmarks. Additionally, we marginalize out landmarks, as shown in Figure 5.4, if the only connected pose is the last one in the sliding window and subject to marginalization.

We divide the set of all states $\boldsymbol{x} = \{\boldsymbol{x}_m, \boldsymbol{x}_n, \boldsymbol{x}_r\}$ into marginalization nodes $\boldsymbol{x}_m$, directly connected neighbor nodes $\boldsymbol{x}_n$, and the remaining states $\boldsymbol{x}_r$. In the following we, call the set $\{\boldsymbol{x}_m, \boldsymbol{x}_n\}$ the *marginalization blanket*. Considering these



(a) marginalization blanket (b) dense marginalization (c) individual global priors

Figure 5.4: Illustration of our approach for computing global priors in order from left to right. All f are factors, whereas $\boldsymbol{x}^p$ are poses and $\boldsymbol{x}^l$ are landmarks in the sliding window graph. (a) The highlighted nodes and factors are part of the marginalization blanket involved in our marginalization and sparsification process. (b) Graph after marginalizing out the oldest pose $\boldsymbol{x}_0^p$. The dense factor f^p represents the marginalized information. (c) The result of our sparsification step. We compute an individual global prior for each remaining state of the marginalization blanket.

definitions, we rewrite the general optimization problem from Equation (4.2) as

$$\begin{aligned}
\boldsymbol{x}^* &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_k \boldsymbol{e}_k^\top \Omega_k \boldsymbol{e}_k \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_j \mathrm{e}_j(\boldsymbol{x}_m, \boldsymbol{x}_n, \boldsymbol{z}_j)^\top \Omega_j \mathrm{e}_j(\boldsymbol{x}_m, \boldsymbol{x}_n, \boldsymbol{z}_j) \\
&\qquad + \sum_{i_{\setminus j}} \mathrm{e}_i(\boldsymbol{x}_r, \boldsymbol{z}_i)^\top \Omega_i \mathrm{e}_i(\boldsymbol{x}_r, \boldsymbol{z}_i) \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} \, \mathrm{F}^{m,n}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right) + \mathrm{F}^{r}\left(\boldsymbol{x}_n, \boldsymbol{x}_r\right), \qquad (5.1)
\end{aligned}$$

where $i_{\setminus j}$ denotes the absence of any measurement connected to the marginalization nodes, $\mathrm{F}^{m,n}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right)$ is the cost of the marginalization blanket, and $\mathrm{F}^{r}\left(\boldsymbol{x}_n, \boldsymbol{x}_r\right)$ captures the cost of the rest of the graph. Following the argumentation of Eckenhoff et al. (2016), we can write Equation (5.1) as a cost function problem

$$\begin{aligned}
C &= \min_{\boldsymbol{x}} \left(\mathrm{F}^{m,n}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right) + \mathrm{F}^{r}\left(\boldsymbol{x}_n, \boldsymbol{x}_r\right)\right) \\
&= \min_{\boldsymbol{x}_n, \boldsymbol{x}_r} \left(\min_{\boldsymbol{x}_m} \mathrm{F}^{m,n}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right) + \mathrm{F}^{r}\left(\boldsymbol{x}_n, \boldsymbol{x}_r\right) \right).
\end{aligned}$$

We minimize the cost of the marginalization blanket $\mathrm{F}^{m,n}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right)$ via Taylor approximation and solve for the optimal marginalization state $\boldsymbol{x}_m^*$ with linearization points $\breve{\boldsymbol{x}}_m$ and $\breve{\boldsymbol{x}}_n$ in the following. The Taylor approximation is

$$\mathrm{F}^{m,n}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right) \approx \mathrm{F}^{m,n}\left(\breve{\boldsymbol{x}}_m, \breve{\boldsymbol{x}}_n\right) + \boldsymbol{b}^\top \Delta\boldsymbol{x} + \frac{1}{2} \Delta\boldsymbol{x}^\top H \Delta\boldsymbol{x}, \qquad (5.2)$$

where the Hessian H, gradient $\boldsymbol{b}$, and $\Delta\boldsymbol{x}$ are defined as

$$H = \begin{bmatrix} H_m & H_{m,n} \\ H_{m,n}^\top & H_n \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_m \\ \boldsymbol{b}_n \end{bmatrix}, \tag{5.3}$$
$$\Delta\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_m - \breve{\boldsymbol{x}}_m \\ \boldsymbol{x}_n - \breve{\boldsymbol{x}}_n \end{bmatrix} = \begin{bmatrix} \Delta\boldsymbol{x}_m \\ \Delta\boldsymbol{x}_n \end{bmatrix}.$$

Solving Equation (5.2) for the optimal marginalization state $\boldsymbol{x}_m^*$ results in

$$\boldsymbol{x}_m^* = \breve{\boldsymbol{x}}_m - H_m^{-1}(\boldsymbol{b}_m + H_{m,n}^\top \Delta\boldsymbol{x}_n). \tag{5.4}$$

By substituting Equation (5.4) back into Equation (5.2), we render the cost of the marginalization blanket independent of the marginalized states $\boldsymbol{x}_m$, such that

$$\begin{aligned} \mathrm{F}^{m,n}(\boldsymbol{x}_m, \boldsymbol{x}_n) &\approx \mathrm{F}^{m,n}(\boldsymbol{x}_n, \breve{\boldsymbol{x}}_n) \\ &= \frac{1}{2}\Delta\boldsymbol{x}_n^\top H_t \Delta\boldsymbol{x}_n + \boldsymbol{b}_t^\top \Delta\boldsymbol{x}_n + c, \\ \text{with} \quad H_t &= H_n - H_{m,n}H_m^{-1}H_{m,n}^\top, \\ \boldsymbol{b}_t &= \boldsymbol{b}_n - H_{m,n}H_m^{-1}\boldsymbol{b}_m, \end{aligned} \tag{5.5}$$

and c is a constant that is omitted when applying the argmax operator. The interpretation behind Equation (5.5) is that we have performed marginalization over the marginalization nodes $\boldsymbol{x}_m$ resulting in

$$p(\boldsymbol{x}_n) = \int p(\boldsymbol{x}_n, \boldsymbol{x}_m)\mathrm{d}\boldsymbol{x}_m = \mathcal{N}\left(\breve{\boldsymbol{x}}_n, H_t^{-1}\right). \tag{5.6}$$

We call the distribution $p(\boldsymbol{x}_n)$ the *marginalization prior*. Compared to using the Schur complement of Equation (5.3) to compute H_t and $\boldsymbol{b}_t$, Equation (5.5) emphasizes the effect of the gradient $\boldsymbol{b}_t$ on the optimization problem. Eckenhoff et al. (2016) argue that the performed marginalization is only optimal if the linearization point $\breve{\boldsymbol{x}}_n$ is the local optimum of the marginalization blanket, and, thus, the gradient $\boldsymbol{b}_t$ vanishes. Extending their argumentation, we show in Section 5.3.5 how to use a linearization point that is not the local optimum.

5.3.4 Sparsifying the marginalization prior

In the following, we show how to derive sparse global priors that approximate dense marginalization in closed form. Our sparsification scheme is designed to have the exact same sparsity pattern compared to using no marginalization. This means that our approach does not suffer from fill-in and simultaneously computes more precise estimates than without using marginalization. As we derive individual global priors for each involved state, we can visually draw them similar to other measurements. This is not only beneficial for data inspection but also helps

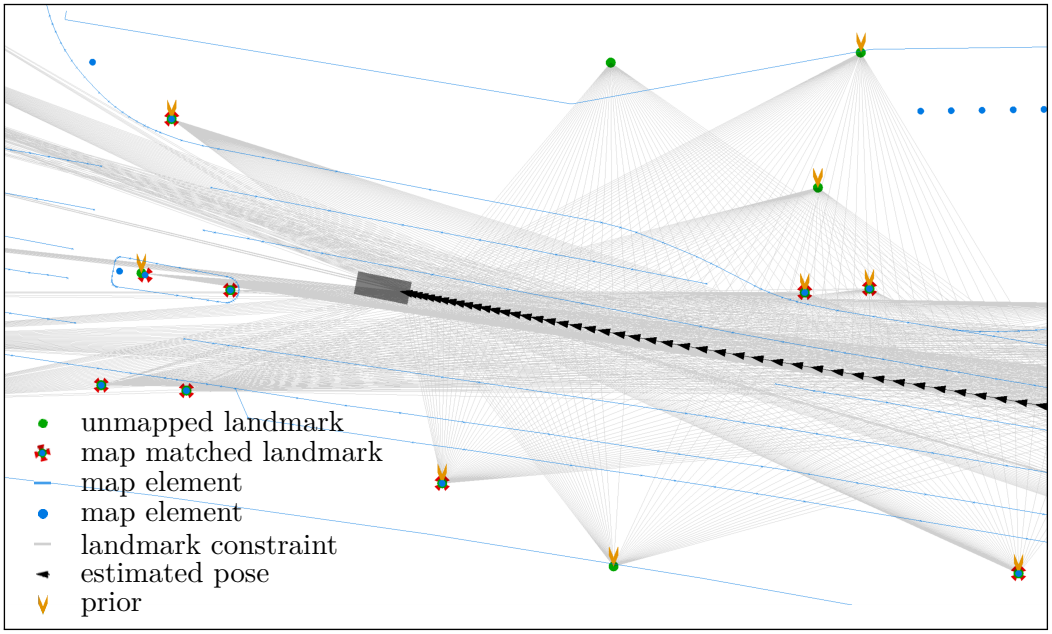


Figure 5.5: The figure shows a screenshot of our implementation with an exemplary real-world sliding window graph that contains sparse global priors. In this example, some landmarks have a sparse global prior constraint while others do not. If a landmark measurement was already removed from the sliding window graph, the corresponding landmark has a sparse prior. The tip of the prior symbol marks the position of the sparse prior. Due to the low measurement noise, the prior positions are nearly identical to the estimated landmark positions.

to understand the effect of our approximation. Figure 5.5 illustrates a real-world example.

Instead of applying the marginalization prior directly in the next sliding window, we first sparsify the distribution $p(\boldsymbol{x}_n)$ to avoid fill-in and ensure sparsity in the graph. By design, we compute a global prior for each state $\boldsymbol{x}_n$ inside the marginalization prior. We formulate the sparsification problem as a minimization of the Kullback-Leibler (KL) divergence $\mathcal{D}_{KL}$ as

$$\begin{aligned} &\underset{\mu_a, \Omega_a}{\operatorname{argmin}} \, \mathcal{D}_{KL}\left(\mathcal{N}\left(\breve{\boldsymbol{x}}_n, H_t^{-1}\right) \parallel \mathcal{N}\left(\mu_a, \Omega_a^{-1}\right)\right), \\ &\qquad \text{with} \quad \mathcal{N}\left(\mu_a, \Omega_a^{-1}\right) = \prod_i \mathcal{N}\left(\mu_{ai}, {\Omega_{ai}}^{-1}\right), \end{aligned}$$

where $\mathcal{N}\left(\mu_{ai}, {\Omega_{ai}}^{-1}\right)$ is the approximated prior for each individual neighbor state in the marginalization blanket. The KL-divergence between the two multivariate normals is defined as

$$\begin{aligned} &\mathcal{D}_{KL}\left(\mathcal{N}\left(\breve{\boldsymbol{x}}_n, H_t^{-1}\right) \parallel \mathcal{N}\left(\mu_a, \Omega_a^{-1}\right)\right) \\ &\quad = \frac{1}{2}\left(\operatorname{tr}\left(\Omega_a H_t^{-1}\right) + \left(\mu_a - \breve{\boldsymbol{x}}_n\right)^\top \Omega_a \left(\mu_a - \breve{\boldsymbol{x}}_n\right) - d + \ln\left(\frac{\det(\Omega_a^{-1})}{\det(H_t^{-1})}\right)\right), \end{aligned}$$

with the dimension d of the state vectors. In the multivariate normal case the KL-divergence reaches its minimum if both means are equal ($\mu_a = \breve{\boldsymbol{x}}_n$). Thus,

the approximation problem reduces to

$$\begin{aligned}\underset{\Omega_a}{\operatorname{argmin}}\, \mathcal{D}_{KL} &= \underset{\Omega_a}{\operatorname{argmin}} \left(\operatorname{tr}(\Omega_a H_t^{-1}) - \ln\left(\det(\Omega_a H_t^{-1})\right)\right) \\ &= \sum_i \left(\operatorname{tr}(\Omega_{ai}) - \ln\left(\det(\Omega_{ai}\{H_t^{-1}\}_i)\right)\right), \end{aligned} \tag{5.7}$$

where we have applied the definition of $\mathcal{N}\left(\mu_{ai}, \Omega_{ai}^{-1}\right)$ and use the notation $\{H_t^{-1}\}_i$ to reference the i-th block-diagonal entry of H_t^{-1}. Deriving Equation (5.7) with regard to Ω_{ai} yields for each individual prior the optimal information matrix

$$\Omega_{ai}^* = \{H_t^{-1}\}_i^{-1}. \tag{5.8}$$

Given our specific topology in the form of absolute priors, Equation (5.8) shows that we can ignore the covariances between states to get the optimal approximation. We denote the sparsified distribution as $\tilde{p}(\boldsymbol{x}_n) = \mathcal{N}\left(\breve{\boldsymbol{x}}_n, \Omega_a^{-1}\right)$, where Ω_a^{-1} is block-tridiagonal. Inserting the approximation into Equation (5.5) yields

$$\mathrm{F}^{m,n}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right) \propto \frac{1}{2}\Delta\boldsymbol{x}_n^\top \Omega_a \Delta\boldsymbol{x}_n + \boldsymbol{b}_t^\top \Delta\boldsymbol{x}_n, \tag{5.9}$$

which we use in the following.

5.3.5 Computing sparse global priors

By extending the proof of Eckenhoff et al. (2016), we now show that we must not necessarily use the local optimum of the marginalization blanket as the linearization point $\breve{\boldsymbol{x}}_n$. Instead, we utilize the previous global estimate by including the gradient term inside the quadratic term. We first note that any cost term independent of the state variable is constant and neglected during optimization. This allows us to add the constant term $c = \Omega_a^{-1}\boldsymbol{b}_t\Omega_a\Omega_a^{-1}\boldsymbol{b}_t$ to Equation (5.9). By rearranging and completing the square, we obtain

$$\begin{aligned}\mathrm{F}^{p}\left(\boldsymbol{x}_n, \breve{\boldsymbol{x}}_n\right) &\propto \frac{1}{2}(\Delta\boldsymbol{x}_n + \Omega_a^{-1}\boldsymbol{b}_t)^\top \Omega_a (\Delta\boldsymbol{x}_n + \Omega_a^{-1}\boldsymbol{b}_t) \\ &= \frac{1}{2}(\boldsymbol{x}_n - \underbrace{(\breve{\boldsymbol{x}}_n - \Omega_a^{-1}\boldsymbol{b}_t)}_{\hat{=}\boldsymbol{\mu}_p})^\top \Omega_a (\boldsymbol{x}_n - \underbrace{(\breve{\boldsymbol{x}}_n - \Omega_a^{-1}\boldsymbol{b}_t)}_{\hat{=}\boldsymbol{\mu}_p}). \end{aligned} \tag{5.10}$$

We refer to the distribution

$$\tilde{p}(\boldsymbol{x}_n) = \mathcal{N}\left(\boldsymbol{\mu}_p, \Omega_a^{-1}\right) = \prod_i \mathcal{N}\left(\boldsymbol{\mu}_{p_i}, \Omega_{a_i}^{-1}\right) \tag{5.11}$$

as the *sparse global prior distribution*. It consists of individual prior terms for each state in $\boldsymbol{x}_n$. By including the gradient term inside the quadratic term, we compensate for the effects of the gradient term without using the local optimum

of the marginalization blanket as the linearization point. Therefore, our approach renders local optimization of the marginalization blanket unnecessary. Returning to our original sliding window optimization problem in Equation (4.2), we include our *sparse global prior distribution* as the marginalization cost

$$\mathrm{F}^{\mathrm{marg}}(\boldsymbol{x}) = \sum_i (\boldsymbol{x}_n - \boldsymbol{\mu}_{p_i})^\top \Omega_{ai} (\boldsymbol{x}_n - \boldsymbol{\mu}_{p_i}),$$

which concludes our derivation.

5.4 Summary

In this chapter, we proposed an approach for delayed map refinement in the context of automated driving. We conceptually discussed integrating a back-end service that is responsible for verifying update hypotheses that it receives from a fleet. Our presented main focus is on reliably estimating the landmark positions in our sliding window graph in a computationally tractable way. Therefore, we extended our graph-based sliding window approach such that we can compute the update hypotheses at runtime. In comparison to related work, we investigated the special case of sliding window graphs for which different assumptions hold compared to arbitrary graphs. We introduced sparse global prior for approximating dense marginalization, which allows us to capture information that we remove from our sliding window in an efficient way. Therefore, we derived how to compute individual global priors for all states involved in marginalization. Additionally, we designed our approach to have exactly the same sparsity pattern compared to using no marginalization such that our graph-based sliding window does not suffer from fill-in during optimization. Furthermore, we presented how to use global linearization so that we do not require an additional local optimization step. In sum, our approach contributes to computing reliable and accurate landmark positions in sliding windows graphs for delayed map refinement.

Chapter 6

Experimental Evaluation

We evaluate our graph-based localization approach with respect to two different use cases. At first, we focus on the capabilities of our system as a pure localization approach on a third-party map. The results of our experiments support the key claims, which are that our graph-based localization approach

(i) provides globally accurate pose estimates,

(ii) incorporates landmark measurements in a generic sensor-independent fashion,

(iii) utilizes general-purpose third-party landmark maps,

(iv) integrates delayed measurements,

(v) revises map associations to increase the localization quality, and

(vi) is fast and frequent enough for application in an automated vehicle.

Additionally, we show that our approach requires GNSS only once for global initialization, is favorable over particle filters for localization, and has a high availability even in challenging urban scenarios. Furthermore, we evaluate our system's capability to compute delayed map refinements in Section 6.6. Our related experiments are designed to show the capabilities of our sparse global prior method and to support our key claims, that our approach

(i) approximates sliding window marginalization with sparse global priors,

(ii) utilizes global linearization points and thus does not require local optimization,

(iii) computes conservative landmark positions for incremental map refinement,

(iv) has exactly the same sparsity pattern compared to using no marginalization but provides more accurate estimates,

(v) is able to improve existing map landmarks.

We start by describing the sensor setup and data recordings before reporting on the results of our experimental evaluation.

6.1 Dataset description

In this section, we present the sensor setup that was used for our data recordings, as well as the odometry module and landmark detectors that we applied. The presented modules are applied within various recordings that we use throughout our evaluation. In the following, we provide a detailed overview of the set of data recordings that we use for evaluating our localization approach on a third-party map within an urban environment. By the time this thesis was written, no urban driving dataset was publicly available for commercial use that provided the required data for our use case. A major benefit of using our own self-recorded dataset is that the estimated trajectories are computed live within the vehicle instead of post-processing the recorded raw data. This ensures that our evaluation reflects the real-world performance of our approach. Our urban driving dataset aims to provide a representative amount of challenging sceneries within a typical urban environment. The dataset was recorded on five subsequent days in December 2019 in Hamburg, Germany, at different times of the day. It contains a wide variety of environmental sceneries, as well as different traffic situations, weather, and lighting conditions. Typical sceneries in the dataset are urban canyons, huge and small traffic junctions, bridges in various styles, and tree-lined roads. Figure 6.1 provides a few representative pictures of the dataset. In numbers, the total driven distance in the dataset is 319 km, which roughly corresponds to 16 h of driving. We provide an overview of the individual trajectories in Figure 6.2. Apart from our Hamburg dataset, which we use to evaluate our localization approach in urban scenarios, we also use further simulated and real-world datasets to evaluate our map refinement approach.

6.1.1 Sensor setup

The prototype vehicle we used in our urban dataset is an e-Golf similar to the one shown in Figure 6.3. Its sensor setup is illustrated in Figure 6.4.

The relevant sensors used as input for our approach are five Velodyne VLP-32C LiDAR sensors (Veloyne Lidar, 2020), a front-view camera, four top-view cameras facing the ground, a low-cost u-blox NEO-M8L GNSS receiver (u-blox, 2020), and an odometry system. The latter is a successor of the approach presented by Baer et al. (2009) and integrates an Xsens MTi 100-series IMU (Vydhyanathan and Bellusci, 2018), wheel ticks, and the steering angle. We provide an

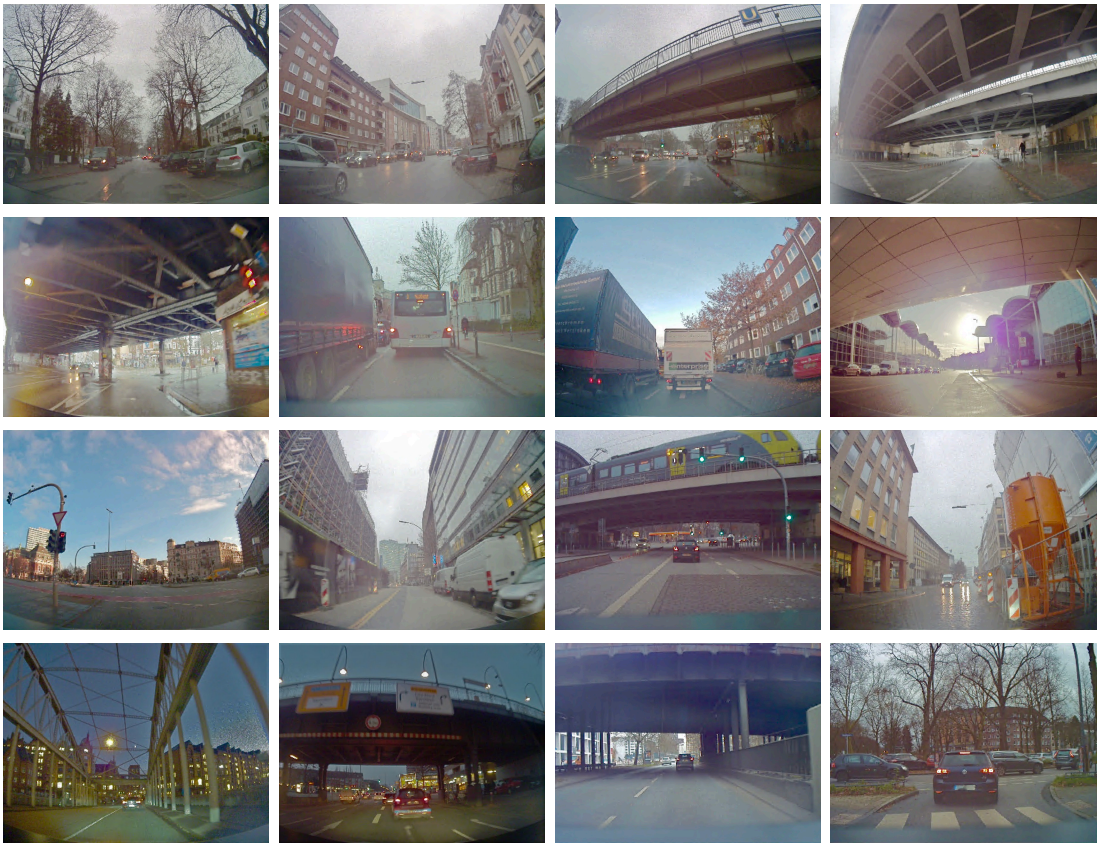
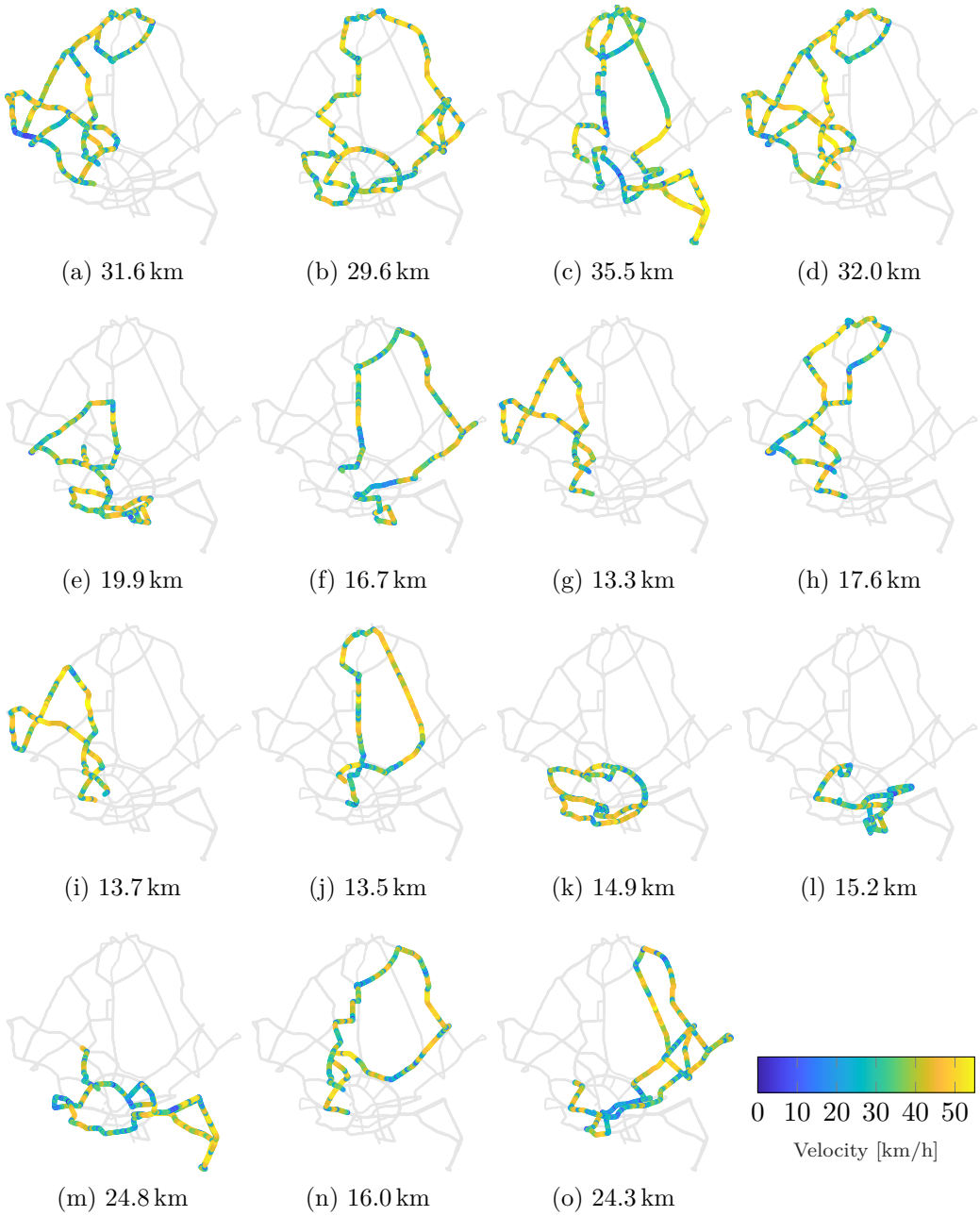


Figure 6.1: An impression on the variety of sceneries within our urban dataset. The dataset contains day and night, sunny and rainy, as well as light and heavy traffic sequences. Typical for our dataset are urban canyons, occasional bridges, and tree-lined roads.



(a) 31.6 km (b) 29.6 km (c) 35.5 km (d) 32.0 km

(e) 19.9 km (f) 16.7 km (g) 13.3 km (h) 17.6 km

(i) 13.7 km (j) 13.5 km (k) 14.9 km (l) 15.2 km

(m) 24.8 km (n) 16.0 km (o) 24.3 km

Figure 6.2: Overview of the different trajectories in the urban dataset with their colored velocity profile. We provide a combined overview on an aerial image in Figure 6.18.

Figure 6.3: The prototype e-Golf used in our experiments. It is equipped with various sensors, as illustrated in Figure 6.4. Source: Volkswagen AG (2019).

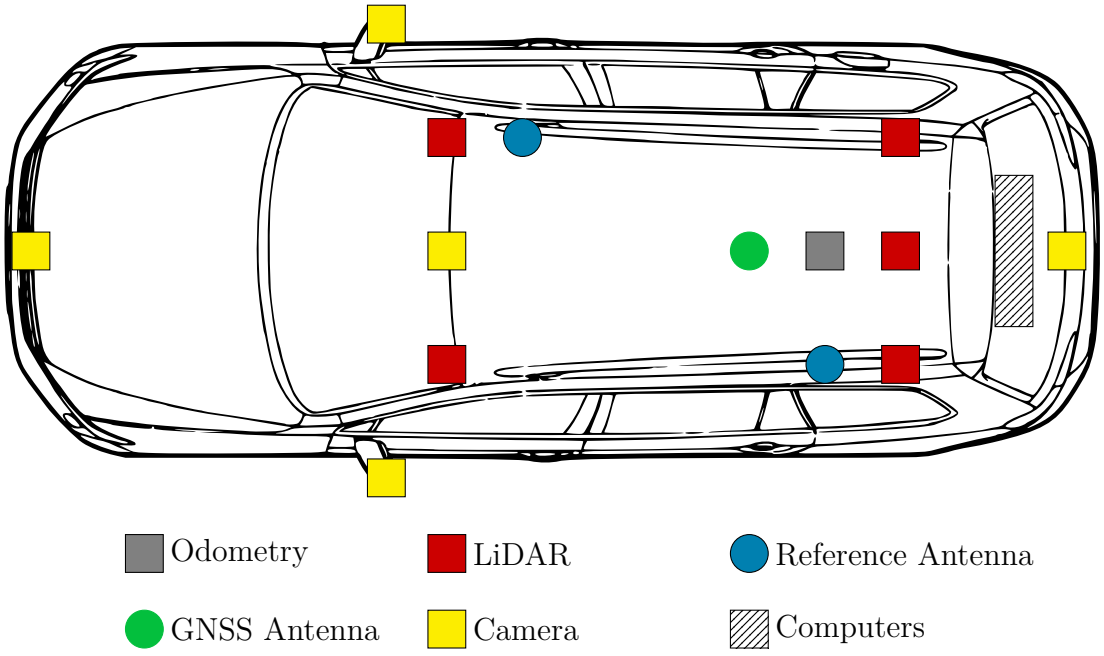


Figure 6.4: The figure illustrates the relevant sensors within the experiments of this thesis. The prototype vehicle is equipped with five Velodyne VLP-32C LiDAR sensors (Veloyne Lidar, 2020), four surround-view cameras, and one forward-facing camera behind the windshield for environmental perception. The low-cost u-blox NEO-M8L GNSS receiver (u-blox, 2020) uses a single antenna, whereas the Applanix LV 520 reference system (Applanix, 2019) and the RT3000 (Oxford Technical Solutions, 2020) use the dual antenna setup. The EgoMaster odometry module (Baer et al., 2009) is a black box system that integrates an Xsens MTi 100-series IMU (Vydhyanathan and Bellusci, 2018), wheel ticks, and the steering angle. All sensors are connected to the computer rack in the trunk of the vehicle. It contains several compute units, which are connected over a network, and run the automated driving stack.

evaluation of the odometry in Section 6.1.2, which gives insights into its accuracy and helps to compare our setup to other approaches. Note that our low-cost u-blox NEO-M8L GNSS receiver does not use dead reckoning, although the receiver supports such an option. It is configured to calculate pose estimates only based on GNSS.

For a comparison with a purely GNSS-based localization approach, we recorded the live trajectories of an RT3000 (Oxford Technical Solutions, 2020). The receiver was equipped with dual antennas and received live correction data. We did not perform any special warm-up procedures for initializing the GNSS receiver. Nevertheless, the system was always started with as much open sky as possible in the urban environment.

As a reference system, we used an Applanix LV 520 (Applanix, 2019) in post-processing mode, which can achieve a 2 cm root mean square (RMS) accuracy and an output frequency of 200 Hz. Its position tracking accuracy without GNSS reception for 60 s degrades to 10 cm. These specifications are achieved with a dual antenna setup, an integrated IMU, GNSS correction data, and a wheel tick encoder. The latter, however, was not part of our setup, which in combination with GNSS outages due to urban canyons and tunnels, partly led to defects in the reference trajectories. As a consequence, we excluded the affected segments from our evaluation so that we can provide an unbiased evaluation. In numbers, this corresponds to 6.61% of the overall recorded data of our urban dataset. Figure 6.5 demonstrates the issue, whereas Table 6.1 contains further information. Note that we never use the RT3000 or the Applanix system as an input to our approach. Instead, we only use the low-cost GNSS receiver.

Recorded driven distance	319.0 km
Maximum velocity	$\leq$ 62.6 km/h
Average velocity	$\sim$ 19.9 km/h
Utilizable driven distance	255.3 km
Excluded due to:	
reference errors	21.1 km
unmapped areas	36.5 km
sensor failure	6.1 km

Table 6.1: Key numbers of our urban dataset. By excluding erroneous parts of the dataset, which are unrelated to our localization approach, we make sure that our evaluation is reliable. Excluded parts contain reference errors, as shown in Figure 6.5, driving in unmapped areas, and sensor failures.

Figure 6.5: The figure shows cases in which the reference trajectory is inaccurate and unusable for our evaluation. This is based on sudden changes, i.e., jumps, in the reference trajectory as, e.g., shown in the bottom right image. Whenever there is a jump in the reference trajectory, the reference system corrects previously accumulated drift. Consequently, the preceding segment of the reference system is inaccurate and thus unusable for our evaluation. We manually identified these problematic segments and excluded the corresponding parts from our evaluation. The trajectory estimates of our system are only shown for visual comparison. Aerial images by Landesbetrieb Geoinformation und Vermessung (2019).

6.1.2 Odometry evaluation

The black box odometry module used in this thesis is a proprietary successor of the EgoMaster (EgM) approach presented by Baer et al. (2009). It combines IMU, wheel-tick, and steering angle measurements to calculate the vehicle's movement with a frequency of 100 Hz. To provide an indication of the odometry module's performance, we evaluated it based on our urban dataset. This may help the reader to put the challenges and results of our urban localization approach into perspective. Additionally, the evaluation allows us to verify the assumption A-6 that the odometry can be considered drift-free within the time span relevant for local association. We found that our odometry module, on average, drifts 24.14 m over 1 km driving within our urban dataset. In this sense, we measure drift as the Euclidean distance between the true vehicle position and the accumulated odometry, when both have the same initial position. Figure 6.6 shows the drift characteristics for different driving distances. Furthermore, we evaluated the odometry based on the KITTI evaluation criteria presented by Geiger et al. (2012). It separates the translational and rotational odometry errors and thus allows to individually analyze both error types. The criteria is used in the popular KITTI visual odometry benchmark[1] to rank various visual odometry ap-

[1] `http://www.cvlibs.net/datasets/kitti/eval_odometry.php`

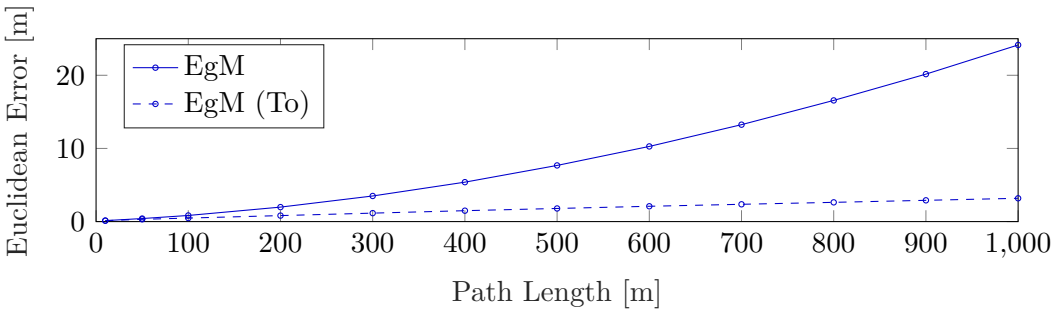


Figure 6.6: Euclidean error of the odometry module that we use as an input for our localization approach. The plot shows the average position offset of the odometry after accumulating it for a specific path length. The curve EgM (To) only considers translational odometry measurements without considering the vehicles heading. On the contrary, the EgM curve includes heading changes, which corresponds to the accumulated full odometry. Comparing both curves shows the impact of rotational odometry errors.

proaches based on the associated KITTI dataset. From this ranking, we choose the leading Vision-lidar Odometry and Mapping (V-LOAM) approach presented by Zhang and Singh (2015), which combines LiDAR and camera-based odometry, and achieves 0.54 % relative translational drift and 0.0013 $\frac{\text{deg}}{\text{m}}$ rotational error on the KITTI dataset. In comparison, the EgM approach, on average, yields 0.48 % translational and 0.0057 $\frac{\text{deg}}{\text{m}}$ rotational error on our urban dataset. Although the results are based on different urban datasets, we consider them comparable w.r.t. their characteristics. This takes into account that both datasets capture a wide variety of urban scenarios such that the results are likely generalizable to other urban areas. We provide more detailed results on the two approaches in Figure 6.7. Although both approaches rely on entirely different sensors, their performance is comparable, while some characteristics are unique to the particular approach. In detail, the plots show that V-LOAM more accurately estimates rotational changes, whereas EgM is superior in estimating translational changes. A characteristic of the EgM approach is that it is more accurate with increasing speed, whereas it is notably worse at low speeds. Judging from outside the black box, this might be an effect of using a wheel-tick sensor, which are comparably unreliable at low speeds. Vice versa, V-LOAM, which does not use a wheel-tick sensor, seems not to have this effect.

Our local association step assumes that the accumulated odometry is considered drift-free for the time span that is covered in the local association (see A-6). In our case, this corresponds to the graph time span $T = 10\,\text{s}$, such that the covered trajectory length depends on the vehicle's velocity, as shown in Figure 6.2. Within our urban dataset, 99.9 % of all graphs are under 83.5 m and on average 15 m long. Considering the odometry drift, as shown in Figure 6.6, the average drift of the accumulated odometry in the local association is roughly

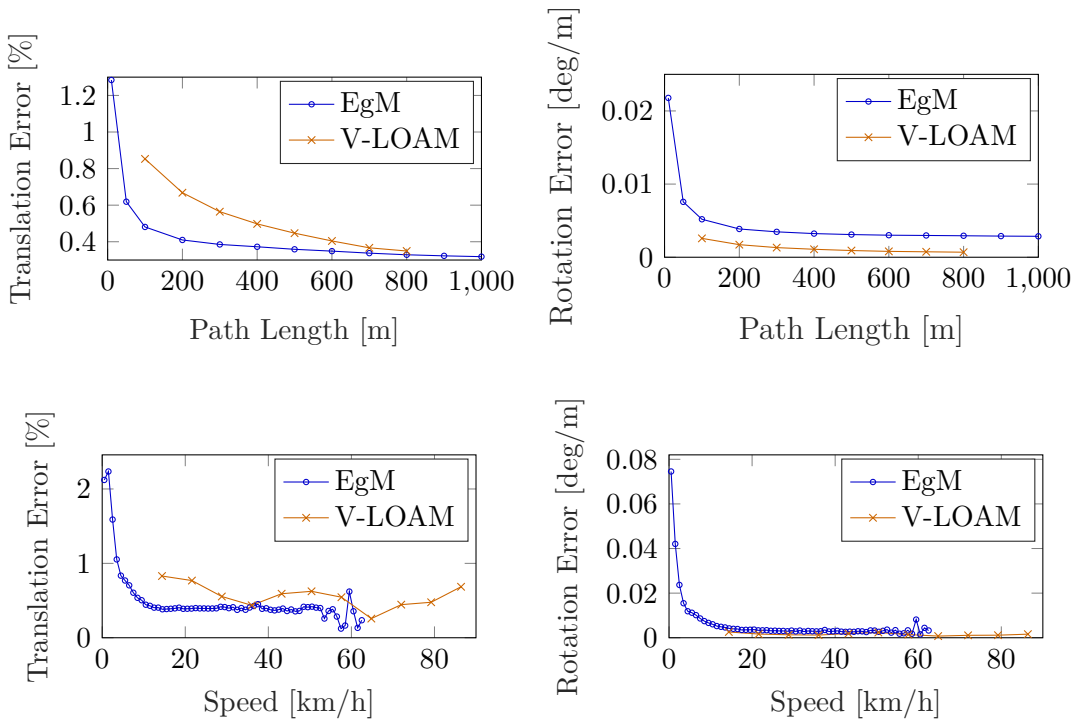


Figure 6.7: Evaluation of the EgM odometry module, which we use as an input for our graph-based localization approach. The shown plots are based on the KITTI odometry evaluation criteria (Geiger et al., 2012). The EgM plots are based on our own urban dataset. As a comparision we show the visual and LiDAR approach V-LOAM by Zhang and Singh (2015) based on the KITTI dataset.

15 cm. However, this includes sliding windows in stopping phases with an effective length of zero and, therefore, negligible drift. We provide a more detailed overview of the accumulated drift in our urban dataset in Figure 6.8. Compared to our landmark detectors' measurement noise, the odometry drift is a negligible minor factor in our local association. We discuss the landmark noise in more detail in the following Section 6.1.3. Even the maximum odometry drift of roughly 85 cm is still acceptable because a single landmark is usually only observed from a section of the accumulated trajectory, which reduces the effective relevant drift.

Overall, we showed that our local association's drift-free odometry assumption (A-6) holds for the EgM odometry module. Furthermore, we demonstrated the quality of the odometry that we use within this thesis. This may ease the comparison of our results to other localization approaches. We conclude that due to the comparable performance, visual odometry approaches might be a promising input alternative to the EgM odometry. This also suggests that our graph-based localization approach is transferable to sensor setups without wheel-tick and IMU sensors.

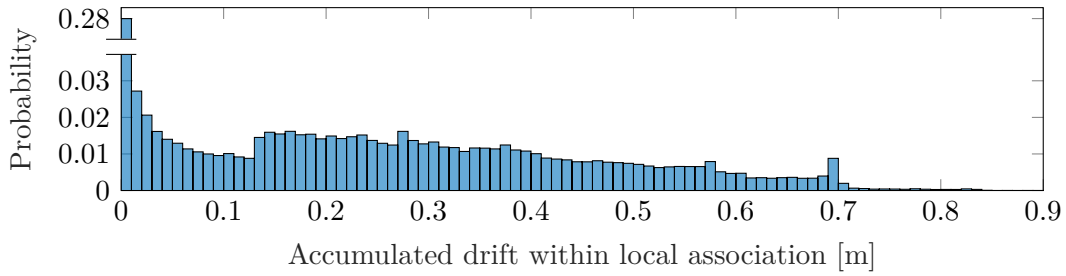


Figure 6.8: Histogram of the odometry drift inherent in our local association within our urban dataset. The drift within our local association depends on the vehicle's velocity within our sliding window and the odometry module's drift characteristics, as shown in Figure 6.6. The most left bin captures 28 % of all occurring drifts within our local association, which is caused by the frequent stop and go traffic in urban areas. Whenever the vehicle stops, the graph's length converges to near zero, such that the sliding window is effectively drift-free. The figure shows that the maximum drift within our urban dataset is roughly 85 cm, which we can consider neglectable for our use case such that our assumption A-6 holds.

6.1.3 Landmark detectors

In the following, we examine the landmark detector modules used in the scope of this thesis. This includes the pole, plane, and lane detectors, available to us as either gray box or black box systems. We exhibit their quality and characteristics to reveal the nature of the input data available to our localization approach in the scope of this evaluation. The number of measurements a detector module provides naturally depends on the number of landmarks in the vehicle's environment. Likewise, the number of received measurements also depends on the processing speed of the underlying sensor and detector module. In our case, the LiDAR-based pole and plane detector processes point clouds with a frequency of 10 Hz, whereas the camera-based detectors are a black box and their internal processing frequency is unknown. Overall, our urban dataset contains 38.13 million pole-, 1.05 million plane-, and 7.61 million lane measurements. Within a time interval of 10 s, which corresponds to our sliding window length, our landmark detectors produce on average 6585 pole, 182 plane, and 1316 lane measurements. In sum, 8083 landmark measurements are on average available to each sliding window graph. After our data association, we incorporate a filtered subset of the landmark measurements into our localization estimate, as explained in Chapter 4.4. In Section 6.2.5, we show that the processing time of our algorithm for all available measurements is sufficient for our algorithm cycle that runs at a frequency of 10 Hz. We provide a more detailed view of the distribution of available measurements in Figure 6.9. Based on the pure amount of measurements, poles are by far the dominating landmark type in our dataset. In combination with the farthest overall detection range, this indicates that pole landmarks significantly impact the results of our evaluation. Due to these properties, pole landmarks are a vital factor in our robust landmark-based urban localization. We show the detection range histograms for the different landmark types used in our urban dataset in Figure 6.10. Furthermore, we demonstrate the spatial distributions within the vehicle reference frame in Figure 6.11. The spatial distributions show that the majority of the pole and plane landmark detections are on the right side of the vehicle. Although one could consider the spatial distributions as, e.g., a weighting during map matching or graph construction, our algorithm considers the landmark detections as uniformly distributed. This ensures that our approach does not overfit to the urban environment of our dataset and can easily be migrated to other environments. After all, our architecture is designed to be a flexible approach instead of deeply integrating specific landmark and sensor properties. In fact, the concepts and algorithms presented in this thesis have also been used by Jürgens et al. (2020), who show the applicability to radar-based odometry and landmark measurements. We provide a visual comparison between using radar-based and LiDAR-based landmarks with our approach in Figure 6.13. As an overview that shows the various

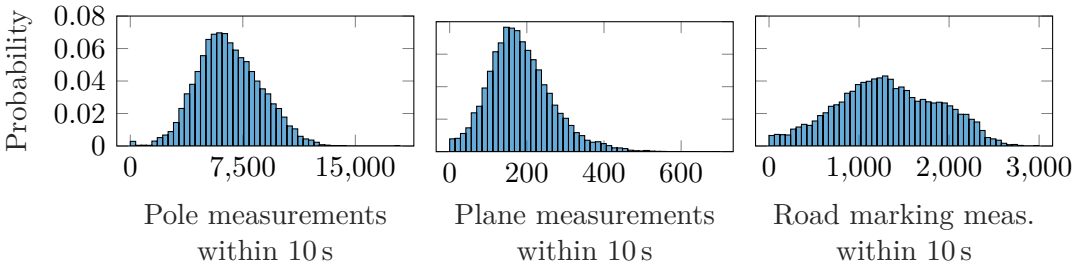
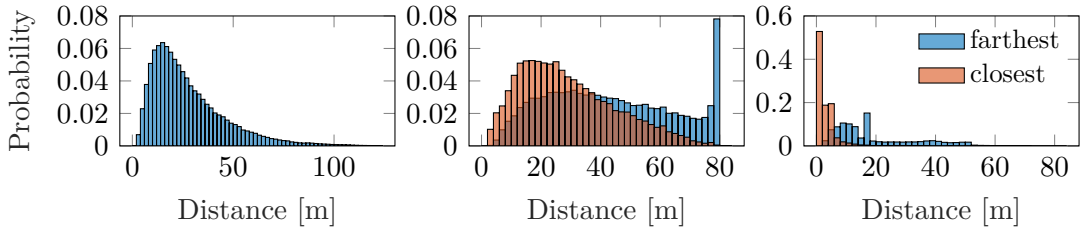


Figure 6.9: Histograms of how many landmark measurements our detectors produce within a 10 s time interval in our urban dataset. The 10 s time interval corresponds to the time span of our sliding window graph and thus reflects how many measurements are processed for each sliding window graph. All shown plots in this figure share the same y axis.

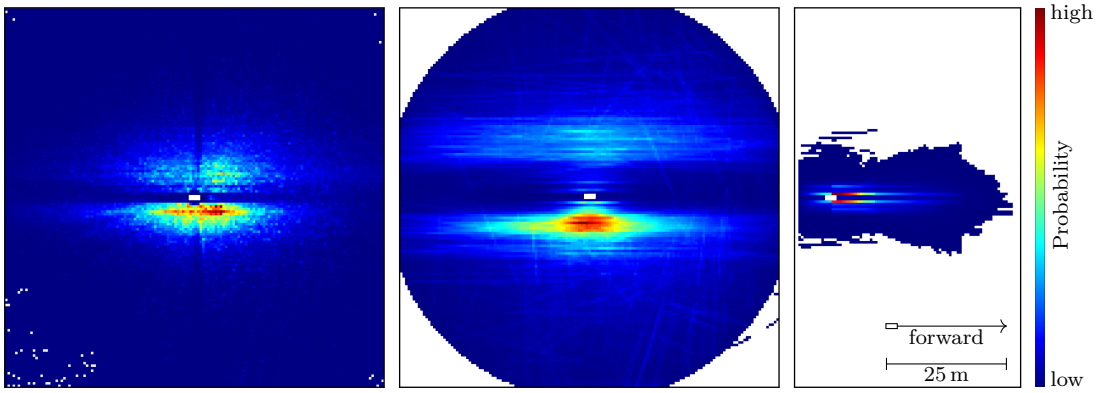


(a) Poles (b) Planes (c) Road markings

Figure 6.10: Detection distances for the different landmark types in our urban dataset. On average, a pole is measured 27.91 m away from the vehicle. For planes and road marking measurements, we depict the histograms for the closest and farthest measurement points to the vehicle. The closest point of a plane measurement is on average 29.14 m away, whereas the farthest one is on average 43.38 m away. For road marking measurements, the average numbers are 3.12 m and 18.83 m. The peak in the histogram for the farthest plane measurements shows that the detector module internally cuts off plane measurements at 80 m.

characteristics of our detectors, Figure 6.12 illustrates raw landmark measurements at on of the junctions within our urban dataset. The figure provides a few examples of false positive detections and shows how well the detections correspond to our third-party map. We discuss the latter and the figure in more detail in Section 6.2.2

In sum, we provided an intuition on the characteristics of the landmark measurements used in the scope of our experimental evaluation. We discussed the spatial, temporal, and range distributions for the pole-, plane- and lane measurements in our dataset, which emphasize the important role of pole-based landmarks in our configuration.



(a) Poles (b) Planes (c) Road markings

Figure 6.11: Spatial distributions of landmark detections within the vehicle reference frameover our complete urban dataset. The vehicle is shown as a white rectangle. White spots in the images indicate that no landmark has been observed for the corresponding coordinate. The shown distributions are discretized to a $1\,\mathrm{m} \times 1\,\mathrm{m}$ grid. The shown spatial area for poles and planes is $75\,\mathrm{m} \times 75\,\mathrm{m}$. The legend in (c) is valid for all images. (a) Pole detections based on LiDAR. We see that the highest probability of detecting a pole is on the right side of the vehicle. This is likely an effect of right-hand traffic, due to which objects on the right side are closer to the vehicle. In combination with decreasing LiDAR resolution over distance, this manifests the shown distribution. (b) Plane detections based on LiDAR. Similar to poles, planes are more likely detected on the right side of the vehicle. The distribution also shows that other vehicles are detected as false positives, which show up in a lane width distance interval parallel to the ego vehicle. Note that we only consider vertical plane measurements, e.g., from building facades, and no ground plane measurements (c) Road marking detections based on top-view and front-facing cameras. The ego lane and parallel lane boundaries are clearly visible. The large spread of the distribution is likely caused by false positive detections.

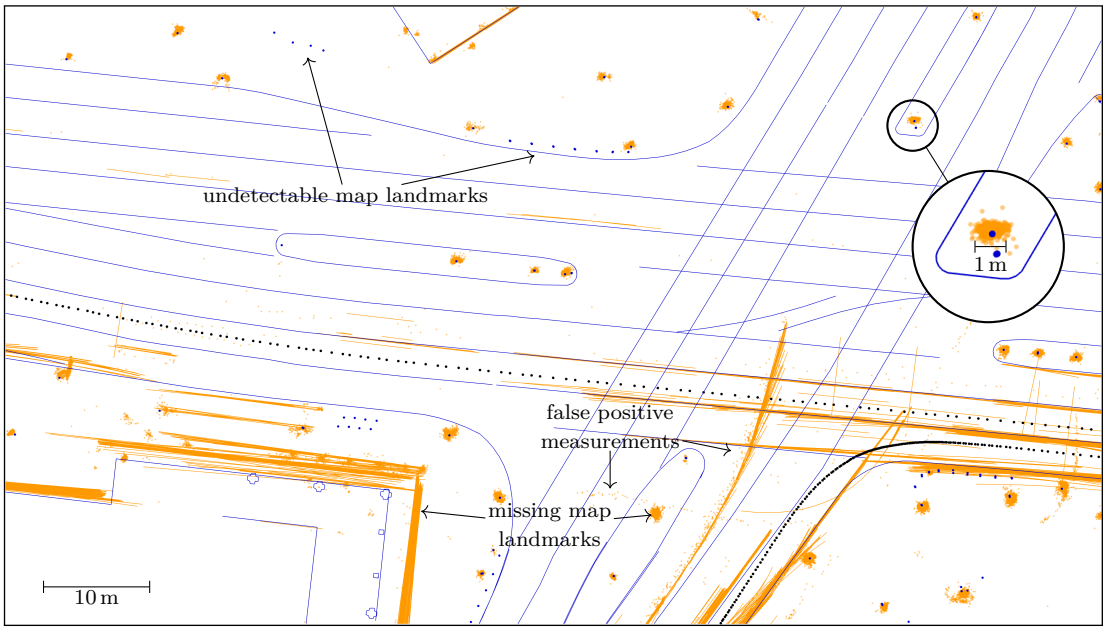
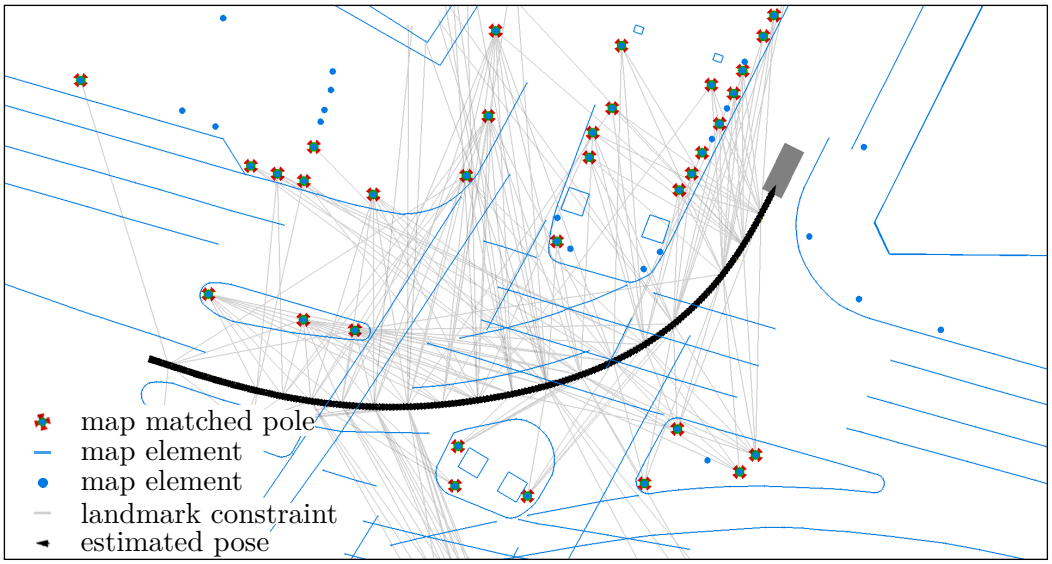
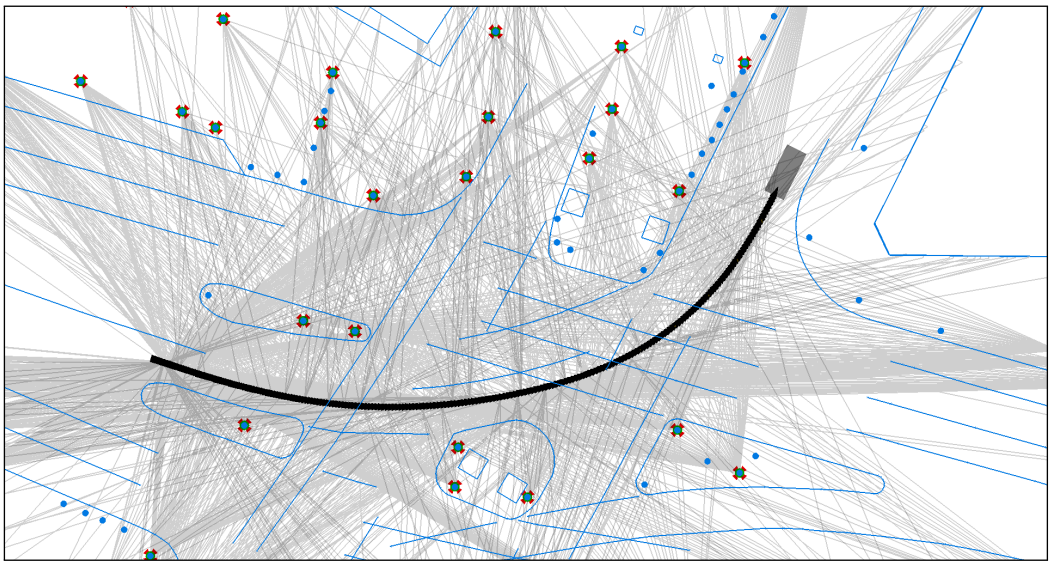


Figure 6.12: Exemplary excerpt of our urban dataset with the raw landmark measurements, in orange, of our pole, plane, and lane detectors. The measurements taken in the vehicle reference frame are attached to the reference trajectories to show them in world coordinates. The vehicle trajectories of two different passes are shown in black, whereas the landmark map is blue. Comparing the map with the measurements illustrates that the measured landmarks only partly coincide with the third-party map. The undetectable map landmarks in the image are short pole landmarks with a small radius, which are in general undetectable with our pole detector and LiDAR sensor combination. In the top right corner of the image, we provide a zoomed-in view of one of the pole landmarks.



(a) Sliding window graph with radar-based landmarks



(b) Sliding window graph with LiDAR-based landmarks

Figure 6.13: The figure shows screenshots of our sliding window graph implementation, which once runs with radar-based landmarks in (a) and once with LiDAR-based landmarks in (b), within our urban Hamburg dataset. In both cases, the additional inputs to our approach are, the third-party map, odometry measurements, and once at startup the low-cost GNSS for global initialization. Note that we never use the RT3000 or the Applanix reference system as an input to our approach. While some of the map landmarks are detected with both sensors, others are only detectable with one of the sensor modalities. This example emphasizes the general-purpose characteristic of the third-party map and also demonstrates the strength of our approach to work with landmark detections from different sensors. In the scope of this thesis, we do not further investigate radar-based landmarks and refer to the work of Jürgens et al. (2020), who apply the concepts of this thesis to radar-based localization.

6.2 Localization on a third-party map

This part of our evaluation focuses on our approach's capabilities to estimate the vehicle pose in urban environments by using a third-party landmark map. We show how well the applied landmark detectors match the deployed third-party map and demonstrate the challenge of using a general-purpose third-party landmark map for localization. At the core of our evaluation, we analyze the accuracy, availability, and computational tractability of our localization based on our urban dataset. An important aspect of this evaluation is that, instead of recording a dataset and afterward post-processing it, our results are based on actual real-world field experiments with live and in-vehicle computed poses. This ensures that the implementation of our localization approach is exposed to perturbations, like measurement delays and time synchronization issues. Therefore, our evaluation reflects the real-world performance and behavior of our approach at runtime in the prototype vehicle. The input to our approach consists of our third-party map, odometry measurements, LiDAR-based and camera-based landmarks, and the low-cost GNSS for global initialization at startup. We never use the RT3000 or the Applanix system as an input to our system and only use it for comparison in the scope of our evaluation. We describe the details of our dataset in the previous Section 6.1.

6.2.1 Key parameters

For the evaluation of our localization approach, we chose to set the applied key parameters of our sliding window approach based on experience and pre-evaluations in an urban environment different from our urban dataset. The sliding window parameters presented in Section 4.2 directly affect the size of the state vector and thus on computation time. Therefore, it is crucial to choose parameters that maximize the time span covered by the sliding window graph while still being computationally tractable. For our urban dataset, we set the number of maximum poses in the graph to $N = 250$, while the frequency of poses inside the graph is $f = 25\,\text{Hz}$. The latter means that all poses within our sliding window graph are by design 40 ms apart. In total, the number of poses and the pose frequency effectively set the total time span covered by the graph to $T = 10\,\text{s}$. We chose an optimization frequency of $f^o = 10\,\text{Hz}$, which in our case is a suitable global pose frequency for automated driving. For our implementation, we rely on g2o (Kümmerle et al., 2011a) as an optimization framework. In this part of our evaluation, we focus on pure localization and rely on graph truncation instead of marginalization and evaluate our approximation of marginalization through sparse global priors w.r.t. map updates separately in Section 6.6. The incorporated landmark types for our urban dataset are poles (i.e., lamp posts, reflector

posts, traffic light posts, pillars, trees, and others) and polyline types consisting of planes and road markings. We chose to use poles exclusively for our transformation matrix search (see Algorithm 3) but include all landmark types for map matching based on the found result. To reduce the computation time required for map matching, we only consider landmarks within a 50 m radius around the initial position guess for the transformation matrices T_{init} during transformation matrix search (see Algorithm 3). Nonetheless, the map matching results proposed in each algorithm cycle are still based on all detected landmarks in the full detection range. Our focus in this experiment is our approach's localization capability, which is why we only include map-matched landmarks in our graph and omit all others within graph optimization.

6.2.2 Incorporating a general-purpose third-party map

In this experiment, we evaluate the challenge of using a general-purpose third-party landmark map for localization and demonstrate that our map matching algorithm is suitable for our localization approach. A main challenge in using general-purpose third-party maps is that the map elements might not well correspond to what the vehicle can detect. On the one hand, a map might contain landmarks that are undetectable by the vehicle's sensor or detectors setup. On the other hand, the vehicle might measure landmarks that are not part of the map. These are either landmarks that are missing in the map or false positive measurements. The magnitude of this map correspondence problem depends on how the map is created. If the mapping vehicle has a similar sensor and detector setup as the vehicle performing localization, the map correspondence problem can be expected to be on a neglectable level. Particularly, if the map is created in a SLAM-based fashion, the landmarks likely correspond well to what the localization vehicle can detect. On the contrary, if the map is created in a black box process, as in our case, the correspondence problem might be significantly more severe. Furthermore, the more time has passed between creating the map and using it for localization, the more outdated the map might be, which worsens the correspondence issue. In our case, the third-party map was created in 2017, whereas the urban dataset was recorded in December 2019. We evaluate the map correspondence problem w.r.t. its magnitude in our urban dataset in the following. Due to missing ground truth data, we consider a heuristic approach for identifying undetectable map landmarks and landmark measurements that are not part of the map. Since the polyline-based landmarks in our third-party map, i.e., planes and lane boundaries, are not stored as single instances, we can not compute reliable and meaningful statistics about their map correspondence. An example of this is a building facade, which, instead of being represented as a single connected polyline, is stored in the map as multiple overlapping plane

instances. Although our detector would measure the building plane and we successfully use it for localization, not each of the single instances might be used. In consequence, our results would depend on how the map is stored and processed and not on its actual representation of the environment. Therefore, we omit polyline landmarks from our map correspondence evaluation. Instead, we focus on pole landmarks, which are clearly defined with a single position. Also, pole landmarks are of interest as their abstract semantic nature makes them prone to correspondence issues. In our case, the challenge is that the map contains trees, traffic light pots, pillars, lamp posts, and others as pole landmarks, while the pole detector just aims to identify cylindrical objects. Our heuristic approach is to project all pole measurements into UTM coordinates by attaching them to the reference trajectory. Afterward, we calculate the Euclidean distance towards the nearest map landmark. Note that the noise level of these globally transformed landmark measurements is based not only on the detectors' noise characteristics but also on the errors within the reference trajectory. For our evaluation, we only consider landmark measurements taken within the mapped areas of our urban dataset. Figure 6.12 shows an example of the globally projected raw landmark measurements of two passes of the same junction. The figure shows several undetectable map landmarks, many false positive measurements, and a few missing map landmarks. Based on our complete urban dataset, we found that 72.13 % (75.05 %) of all globally projected pole measurements are within a 1 m (2 m) radius of a map landmark. Thus, 27.78 % (25.95 %) of all pole measurements are either false positive measurements, are too noisy, or correspond to missing or shifted map landmarks. Based on visual data inspection, most of the false positive measurements seem to stem from dynamic objects, which appears to be a weakness of the given black box pole detector. For example, Figure 6.12 illustrates false positive pole measurements within lanes that likely correspond to pedestrians or other vehicles. The missing map landmarks mostly stem from constructional measures or were missed during initial mapping. Due to missing ground truth data, we can not provide further reliable statistics on missing map landmarks. Another correspondence issue with third-party maps is that not necessarily all map landmarks are detectable by the vehicle. Instead of missed detections due to blocked line-of-sight, we here refer to landmarks that can not be detected with a specific sensor and detector module combination in general. An example in our case are poles with a small diameter and a low height, which return insufficient LiDAR measurements so that our detector can not to detect them. We consider a pole map landmark as detectable if at least ten globally projected measurements are within a 1 m radius around the landmark. Using ten measurements as a boundary allows us to compensate for false positive detections near map landmarks. Comparably, a detectable pole map landmark in our

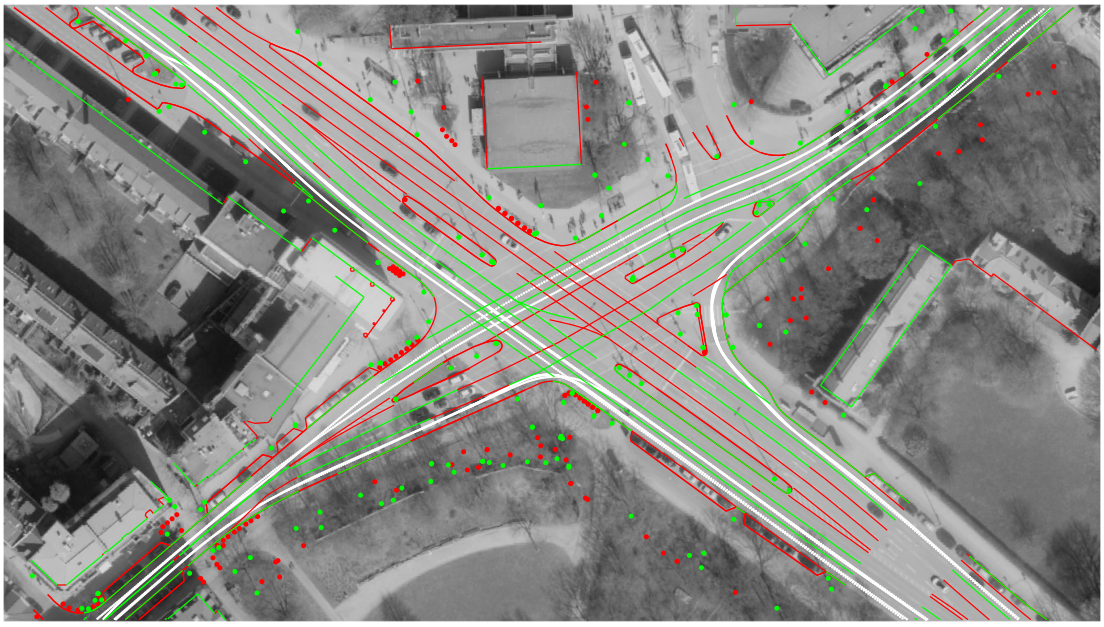


Figure 6.14: Landmarks of the third-party map at an exemplary junction of our urban dataset. The coloring illustrates if a map landmark was detected and matched at least once during multiple passes of the junction. If a landmark was detected and matched to the map, it is shown in green. Otherwise it is colored red. The driven trajectories are shown in white. The reasons for an unmatched map landmark are manifold, which we show in more detail in Figure 6.12. Aerial image by Landesbetrieb Geoinformation und Vermessung (2019).

dataset is observed 513 times on average. We found that within the visited areas of our urban dataset, 53.26 % of all pole landmarks are undetectable, such that only 46.74 % are usable for localization. Even if we would consider a single measurement as sufficient for classifying a pole landmark as detectable, still 41.35 % of all pole landmarks are not detectable with our LiDAR sensor and detector module combination. Overall, this emphasizes the imposed challenge on our map matching algorithm, which at runtime neither has a priori information about a landmarks detectability nor missing map landmarks. Although it might be possible to learn the map correspondence over time, we are in the scope of this thesis interested in using a general-purpose third-party map without further modification. Therefore, we designed our map matching algorithm to cope with a large number of undetectable map landmarks and a high number of false positives. Our localization approach matched and incorporated 44.77 % (21289 of 47556) of all map poles in the third-party map. This is only 1.97 % lower than the 46.74 % limit, which we deduced as being detectable by the prototype vehicle. Figure 6.14 shows these matching results for multiple passes of an exemplary junction.

In sum, we infer that our map matching algorithm successfully copes with false positive measurements, missing map landmarks, and the high number of undetectable map landmarks. Our evaluation w.r.t. localization accuracy (see Section 6.2.3) further supports that our map matching algorithm successfully

works, and we can compensate for the map correspondence issues with our localization approach.

6.2.3 Localization accuracy

This experiment is designed to investigate the accuracy of our approach regarding pose estimates. We present the accuracy of our graph-based sliding window approach in our real-world urban dataset and compare it against a low-cost u-blox NEO-M8L GNSS receiver without dead reckoning (u-blox, 2020) and an RT3000 (Oxford Technical Solutions, 2020). As discussed in Section 6.1.1, we use an Applanix LV 520 system (Applanix, 2019) as a reference and exclude the parts of the dataset in which the reference system fails. To compute the error statistics, we interpolate the 200 Hz reference trajectory to the timestamps of each localization approach. Because of the high-frequency reference trajectory, the time spans that we need to interpolate for our comparisons are minimal. In return, the induced interpolation errors impacting the accuracy of the interpolated reference poses are minimal. For our approach, we use the most recent pose of each optimized sliding window graph as the pose for comparison. This corresponds to the output pose of our approach to other automated driving software modules. We investigate the average Euclidean error as the main measure for the vehicle's position accuracy and separately provide the average heading error. Also, we subdivide the position accuracy into average absolute lateral and average absolute longitudinal errors, which allows a more specific interpretation of our results. For example, in a lane-keeping function, the lateral error is more critical than the longitudinal error. However, for automated driving, we consider the Euclidean error as the main performance indicator. Table 6.2 shows our results for all three localization systems compared to the reference. Moreover, Figure 6.15 shows the error distributions of our graph-based sliding window approach. Within our urban dataset, the low-cost GNSS yields an average Euclidean error of 10.15 m, whereas the RT3000 achieves 0.65 m. In comparison, our approach yields an average Euclidean error of 0.11 m and outperforms the low-cost GNSS receiver and the RT3000. Although RTK-based systems, like the RT3000, may have accuracy specifications up to 0.01 m, they suffer from extended GNSS outages. This is also shown and discussed by Štern and Kos (2018), Joubert et al. (2020), and Reid et al. (2019b). We further discuss the outages of all systems w.r.t. our urban dataset in Section 6.2.4. Given that our reference system operates without a wheel tick encoder and thus does not work under optimal conditions, we suspect that our localization's accuracy might even be better than what our experiment suggests.

In sum, our experiment emphasizes the advantage of our landmark-based localization over GNSS-based systems in challenging urban environments and

supports our claim that our graph-based sliding window approach yields highly accurate pose estimates.

6.2.4 Outages and availability

This experiment is designed to evaluate if our graph-based sliding window approach can provide pose estimates with high availability. We compare the availability of our approach to an RT3000 (Oxford Technical Solutions, 2020) and additionally include a low-cost GNSS receiver in our evaluation. The latter is configured to work without dead reckoning such that it only outputs a pose estimate if sufficiently enough satellites are visible. We use it here to point out the challenging conditions in urban environments for GNSS-based systems. First, we discuss the impact of GNSS outages. A challenge for classical GNSS-based localization approaches are blocked line-of-sight scenarios, in which satellite signal reception is limited or not possible at all. Typically, these occur in urban canyons, tunnels, under bridges, and even on tree-lined roads. A major issue in these scenarios is that low-cost GNSS receivers without dead reckoning fail to compute a pose estimate during the outage, which results in a localization outage for the vehicle. Note that our u-blox NEO-M8L GNSS receiver does not use dead reckoning, although the receiver supports such an option. It is configured to calculate pose estimates only based on GNSS. Based on the low-cost GNSS receivers frequency of 1 Hz, we consider it an outage in the scope of this thesis if a localization system can not provide a vehicle pose for more than two seconds. We provide the outage statistics of our low-cost GNSS receiver for our urban dataset in Figure 6.17. Additionally, we illustrate the GNSS outages that last over ten seconds in Figure 6.18a. These extended outages sum up to roughly 18 % of the overall recorded timestamps. In relation to our graph-based localization approach, for which we choose a ten-second sliding window (see Section 6.2.1), these outages correspond to the sliding window graphs for which no GNSS data is available. Beneficially, our graph-based localization approach only requires GNSS for initialization and thus is unaffected by the GNSS outages. Additionally, we include GNSS as a fallback whenever the vehicle leaves its specified operational domain by driving into an unmapped area. We consider these cases as out-of-operation phases in which the vehicle's pose is only estimated based on odometry and GNSS. Whenever the vehicle reenters the mapped area during an extended GNSS outage, the reinitialization might be delayed until a valid GNSS pose is received. This depends on how far away the vehicle estimates are from their true position during the outage phase. Although our approach perfectly manages to provide vehicle estimates outside of mapped areas, its accuracy reduces to what the GNSS receiver can provide. With respect to the safety in automated driving, we consider it crucial that the vehicle only operates in its specified operational

error type	GBL	low-cost GNSS	RT3000
lateral	0.06 m / (0.05 m)	7.08 m / (4.44 m)	0.43 m / (0.13 m)
longitudinal	0.08 m / (0.06 m)	5.56 m / (3.12 m)	0.39 m / (0.13 m)
heading	0.11 deg/ (0.08 deg)	31.36 deg/ (3.96 deg)	1.05 deg/ (1.07 deg)
Euclidean	0.11 m / (0.09 m)	10.12 m / (7.01 m)	0.65 m / (0.25 m)

Table 6.2: Error statistics of our graph-based localization (GBL) for the urban dataset compared to a low-cost u-blox NEO-M8L GNSS receiver without dead reckoning and an RT3000 system. Our GBL approach only uses the low-cost GNSS once at start up for global initialization and otherwise does not incorporate any GNSS data while driving in a mapped area. All values are stated as mean / (median) of the absolute error values and computed against an Applanix LV 520 reference system that operates without a wheel tick encoder.

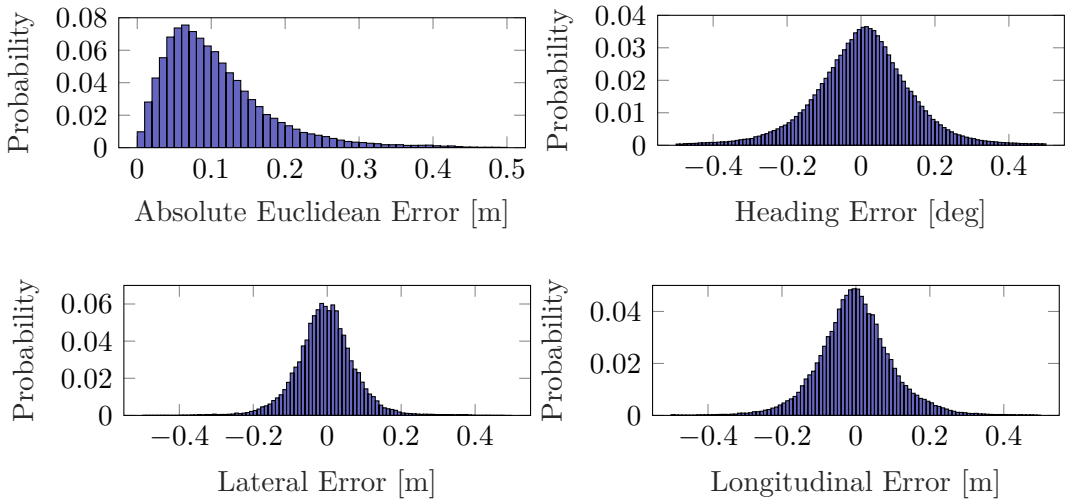


Figure 6.15: Error distributions of our graph-based localization approach for our urban dataset.

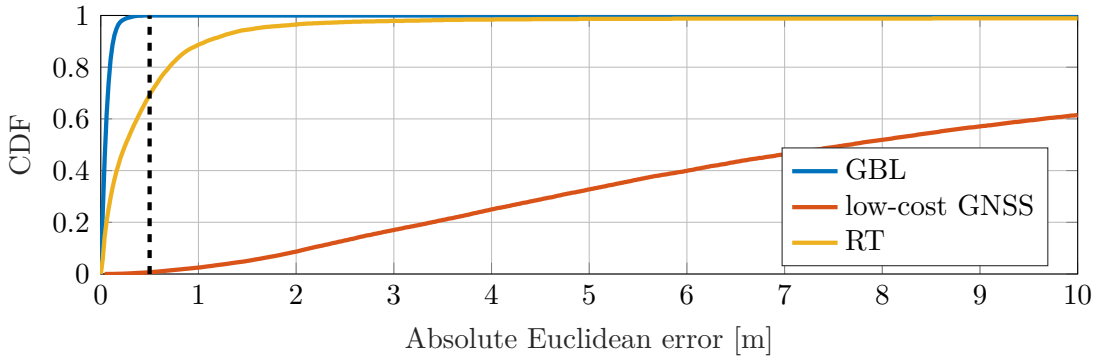


Figure 6.16: Cumulative distribution functions (CDF) for the absolute Euclidean error in our urban dataset for our approach (GBL), RT3000 (RT), and the low-cost GNSS module. In this figure, we assume an example boundary of 0.5 m for the required availability. With respect to this boundary, the RT3000 has a 69.33 % availability, whereas our sliding window approach yields 99.97 % availability.

domain. Thus, we do not further investigate localization in unmapped areas as a part of this thesis.

Apart from system outages, the availability of a localization system is regularly described as the percentage for which the system's accuracy is below a specific error threshold. Which availability boundary is required for automated driving is still an open question in the community. In practice, the influencing factors are not only the operational design domain but also the capabilities, and architecture of the autonomous system itself. This is discussed in more detail by Reid et al. (2019a). Based on our experience, we consider an exemplary Euclidean error boundary of 0.5 m as the availability boundary for our automated system. For this boundary, the low-cost receiver has a near-zero availability, whereas the RT3000 yields a 69.33 % availability. This shows how challenging the urban environment is for GNSS-based approaches. Similar to the low-cost receiver, the GNSS outages seem to have a drastic effect on the RT3000 as well. Although fusing IMU data allows it to continue operating and does not suffer from any outages, its performance suffers under the limited satellite reception. For 29.84 % of the recorded data the RT3000 operates in the highly precise RTK Integer mode, 13.36 % in the less precise, RTK Float mode, 53.1 % in Differential GPS (DGPS) mode, and roughly 3 % without any satellite reception. Compared to the outages of the low-cost receiver, the RT3000 benefits from its dual antenna setup. Our landmark-based approach does not suffer from GNSS outages and achieves 99.97 % availability w.r.t. the 0.5 m boundary within the mapped areas of our urban dataset. We compare the cumulative error distributions of the low-cost GNSS receiver, RT3000, and our sliding window approach within the mapped areas of our urban dataset in Figure 6.16.

Overall, our evaluation w.r.t. availability demonstrates the advantages of our landmark-based localization approach over GNSS. Compared to GNSS-based localization, our approach has a high availability even in challenging urban areas with limited satellite reception, which is a requirement for automated driving applications.

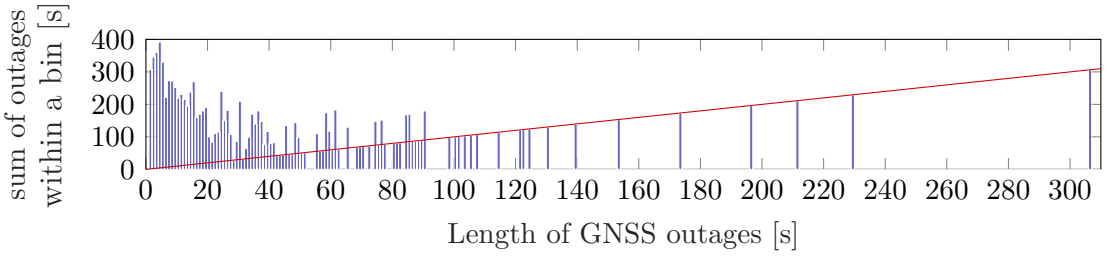


Figure 6.17: Periods of GNSS outages within our urban dataset. The figure shows a histogram with a bin width of one second on the x-axis and the total sum of all individual outages for the specific bin on the y-axis. The red line illustrates the single occurrence boundary, showing that short outages are much more likely than extended outage periods. The maximum outage within the dataset is 306 s, which occurred one time within our dataset.

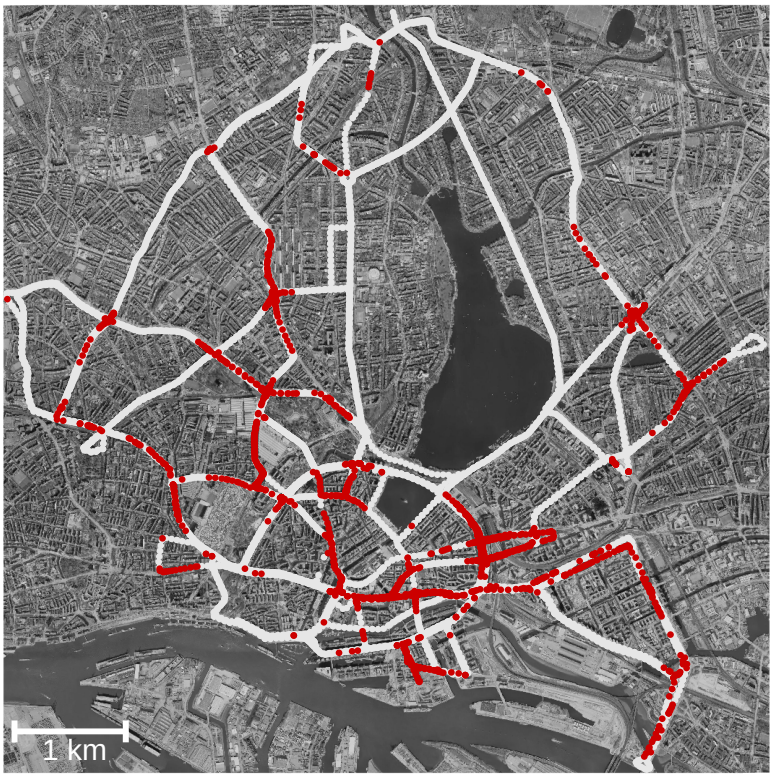
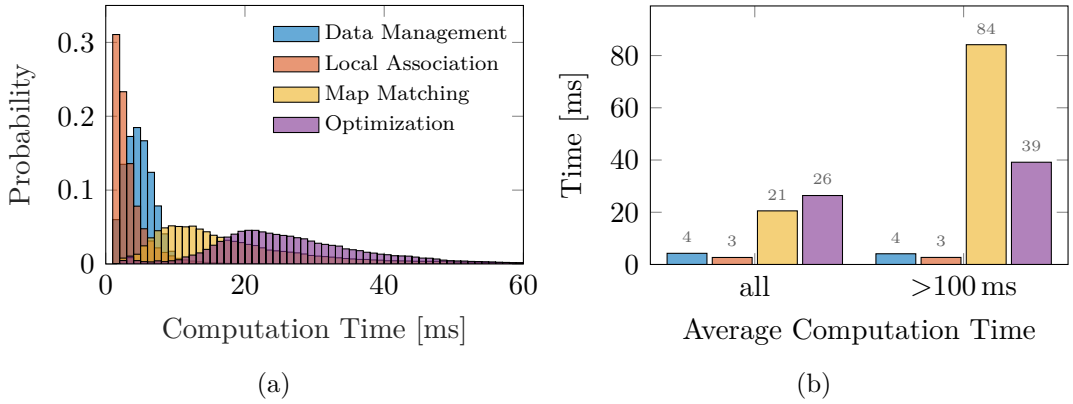


Figure 6.18: GNSS outages over 10 s. The parts of the trajectories for which the low-cost GNSS receiver could compute a pose estimate within ten seconds are shown in white. Vice versa, the locations in our dataset for which the low-cost GNSS receiver was not able to compute a pose estimate for over ten seconds are shown in red. These outages mainly occur in areas with blocked line-of-sight like tunnels, bridges, and urban canyons. Overall, roughly 18 % of the overall recorded trajectories are within a GNSS outage phase. Aerial images by Landesbetrieb Geoinformation und Vermessung (2019).



(a) (b)

Figure 6.19: Computation times for the different steps of our algorithm in our urban dataset. (a) Histograms for the computation times of the different steps of our algorithm. For clarity, the histograms do not show outliers above 60 ms, which have a near-zero probability. (b) Comparison of the different average computation times required by our algorithm. On the left side, we show the average computation times for the different computation steps. On the right side, we illustrate the average numbers for the vehicle poses with an overall computation time over 100 ms, which corresponds to 4.4 % of all computed poses.

6.2.5 Runtime

In the following, we investigate the different influencing factors on the computation time required for each algorithm cycle of our localization approach. We support our claim that our system is fast enough for usage in a prototype vehicle, which requires a pose update every 100 ms on average for automated driving. First, we discuss the computation time and different steps of our algorithm in general. Afterward, we consider the different steps of our algorithm in more detail. Our experiment is based on our urban dataset, in which we limit the number of poses inside the sliding window to 250 with a resolution of 25 Hz. In total, our sliding window graph covers a time span of 10 s and integrates the measurements taken within this time frame. The number of landmarks in our sliding window is unlimited but is, in practice, bounded by the structure of the environment.

Figure 6.19 illustrates the required computation times for the different steps of our algorithm in our urban dataset. Comparing the histograms in Figure 6.19a shows that our algorithm's most time-consuming step is graph optimization followed by map matching. At the same time, the computation time required for locally associating detected landmarks and general data management have a neglectable influence on the overall computation time of an algorithm cycle. In total, our approach achieved an average computation time per algorithm cycle of 54 ms. In detail, 95.6 % of all poses are computed in under 100 ms, whereas only 4.4 % required more computation time. We illustrate the required computation times for the different steps of our algorithm once based on all algorithm cycles and once only based on the algorithm cycles that required more than 100 ms in Fig-

ure 6.19b. In the latter case, our experiment shows that the dominating factor influencing the overall computation time is our map matching. This occurs in situations with many detectable landmarks in the vehicles line-of-sight, as, for example, in large traffic junctions with an unblocked field of view. Since the computation time required by our map matching depends on the number of detectable landmarks, the overall computation time of an algorithm cycle increases accordingly. We found that the increased computation time is not an issue for automated driving in our prototype vehicle. The highly accurate poses of our localization approaches and the high-frequency odometry allow an odometry-based extrapolation of the last computed global vehicle pose so that delay can be compensated. Within our urban dataset, we found that for all vehicle poses that took longer than 100 ms to compute, the maximum driven distance that needs to be extrapolated is 2.8 m. Taking our odometry evaluation in Section 6.1.2 into account, the added odometry drift error is roughly 0.04 m, which we consider acceptable for our use case. Alternatively, to reduce the computation time, we could have limited the number of processed transformation matrix candidates in our map matching or the range threshold for included landmark detections, which in our case is 50 m. Considering that delayed measurements are easily integrated into our sliding window graph, it also seems promising to decouple the map matching from the overall algorithm cycle. Map matching could be processed in its own threaded cycle such that the results are added as delayed constraints to the graph as soon as they are available. Since our presented results already fulfill the requirements on computation time for our prototype vehicle, we did not further investigate these improvements within this thesis's scope.

In the following, we investigate the relation between our map matching algorithm presented in Chapter 4.4.2 and its required computation time in more detail. In the scope of our experimental evaluation, we only consider pole landmarks for determining the best transformation matrix between detected landmarks and third-party map. Therefore, the runtime of our map matching depends on the number of pole clusters that we identified during local association and the number of map poles in the vehicle's environment. On average, our map matching takes 21 ms in our urban dataset, which is, in general, suitable for our use case. In cases with an overall computation time above 100 ms, our map matching takes, on average, 84 ms. In some rare cases, the required time for map matching reached roughly 400 ms (see Figure 6.20). This was caused by a combination of many false positive pole detections in an area with many nearby map landmarks leading to an excessive number of transformation candidates. We do not consider this relevant for the overall interpretation of our results since this issue is specific to our pole detector and implementation. The input to our map matching algorithm in each algorithm cycle in our urban dataset consists, on average, of 88

map poles and 59 pole clusters from our local association. Figure 6.20b illustrates that the number of detected poles exponentially influences the required time for map matching. Nevertheless, the figure also shows that the distribution over the number of considered pole clusters is in practice within a feasible range. Besides, Figure 6.20a indicates a near-linear dependency between the number of considered map poles and the required computation time for map matching. Although we consider map poles within a nested loop, the near-linear dependency results from pre-filtering potential map landmarks for each cluster (see Algorithm 3, line 5). In combination, map poles and clusters yield, on average, 316 transformation candidates between our local map and the third-party map that we assess in each algorithm cycle. Due to the discussed exponential characteristics of Figure 6.20b, the number of transformation candidates exponentially impacts the map matching runtime, shown in Figure 6.20c. Note that we only consider pole clusters and map landmarks within a specified 50 m radius for finding the best transformation. In sum, finding the best transformation between our local map and the third-party map requires, on average, 1.2 million distance calculations between detected and map landmarks (see Algorithm 3, line 14). We refer to the latter as the number of comparisons in the following. Figure 6.20d illustrates the relation between map matching runtime and comparisons, which is linear and depends on the computer's processing power. In our case, our system can compute roughly 12 million comparisons within 100 ms. In sum, our experiments show that the main factors impacting the map matching computation time are the number of transformation candidates assessed in each algorithm cycle and the number of detected poles in our local map. Additionally, we showed that our map matching approach is applicable for online localization in an automated vehicle. It can handle the number of map landmarks and detected landmarks that naturally occur in our urban dataset.

Turning our attention to graph optimization, Figure 6.21 illustrates the relation between optimization time and the number of constraints in our sliding window graph with a time span of 10 s in our experiment. On average, our sliding window graph contains 2996 constraints, from which the majority stems from pole measurements. Figure 6.21a illustrates the empirical distribution over the number of constraints in more detail. Similarly, Figure 6.21b illustrates the empirical distribution over the number of pole vertices in our sliding window graph. On average, our graph contains 42 pole vertices. The figures show that the number of pole constraints and number of pole vertices have a crucial impact on the required optimization time. Since we include poles directly in our state vector, any constraint between a vehicle pose and a pole state produces an off-diagonal entry in the Gauss-Newtons system matrix H (see Section 3.4.2 and Figure 3.8), which increases the time required for the matrix inversion step dur-

ing Gauss-Newton optimization and thus increases optimization time. Likewise, adding more poles to the graph increases the size of the system matrix H, which also influences the required optimization time. Therefore, limiting the number of constraints and limiting the number of poles in the graph are both viable options for controlling the required optimization time. We implicitly do this by setting the number of poses and their temporal distance, which controls the time span that our sliding window covers. We cover the effects of the sliding window length in more detail in the following Section 6.3.1.

An option that would further reduce the optimization time is to rely on pose graphs, which only maintain vehicle poses and do not include any landmarks in their state vector. Consequently, the system matrix H does not contain off-tridiagonal entries such that the optimization can be performed more efficiently. While this is common in related work, e.g., Wu et al. (2017), Harr et al. (2018), and Lategahn et al. (2013), our approach does the contrary. Our main advantages of re-estimating the landmark positions in each algorithm cycle are reduced linearization errors and that it enables us to perform map refinement. As our experiments have shown, the optimization time required for our sliding window approach, despite optimizing landmark states, is suitable for our urban use case.

In sum, our runtime experiments have shown that our graph-based sliding window approach is applicable for automated driving in urban areas as it provides fast and frequent pose estimates.

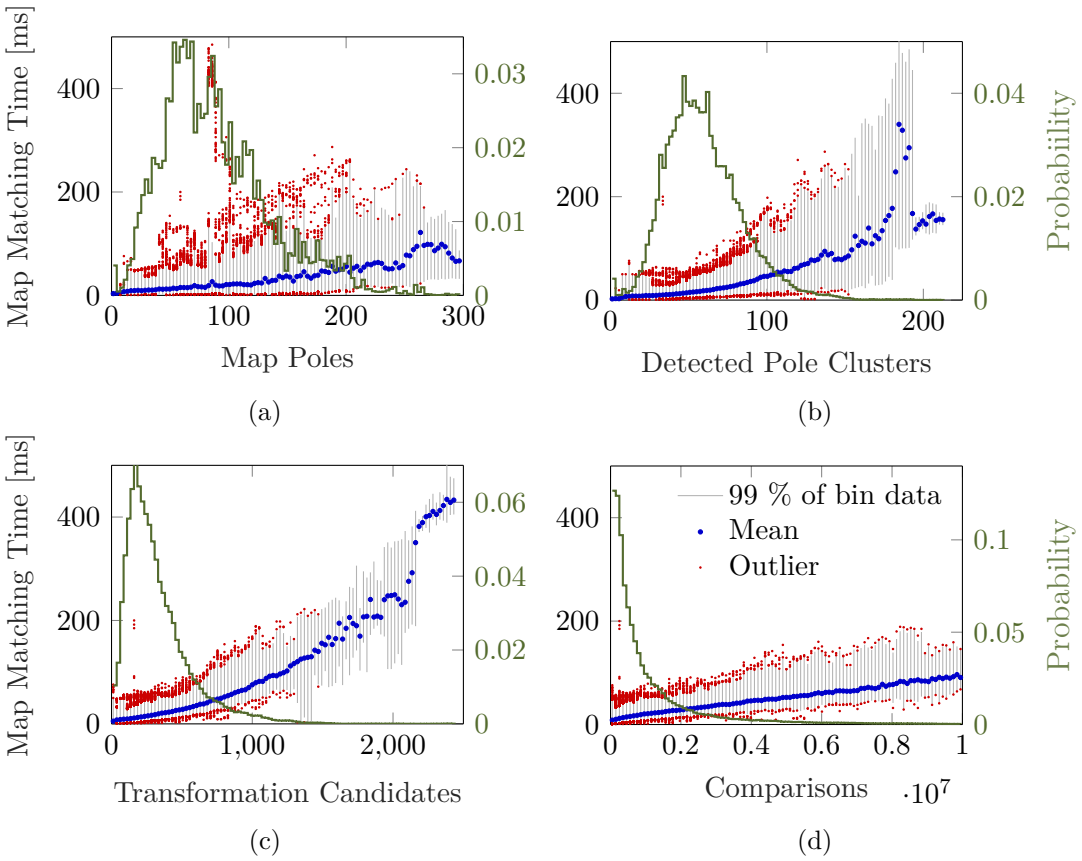


Figure 6.20: Correlation between the computation time required for our map matching (Algorithm 3) and its input data in our urban dataset. All plots illustrate the correlation between the input data on the x-axis and the required computation time for our map matching on the left y-axis. At the same time, the plots illustrate the empirical probability distribution on the right y-axis, which allows to infer the most relevant sections on the x-axis. The higher the probability, the more meaningful is the corresponding section on the x-axis. For all plots, we discretized the x-axis into small bins and computed the mean, which is shown in blue. The vertical gray lines denote the regions covering 99 % of the y values of each bin. All points outside 99 % of each bin are denoted as outliers in red. (a) Relation between the number of considered map poles and map matching time. (b) Relation between the considered poles from our local association step and map matching time. (c) Relation between transformation candidates assessed in each algorithm cycle and map matching time. (d) Number of comparisons between map landmarks and detected landmarks in one algorithm cycle. For clarity, the plot omits outliers requiring over 10 million comparisons.

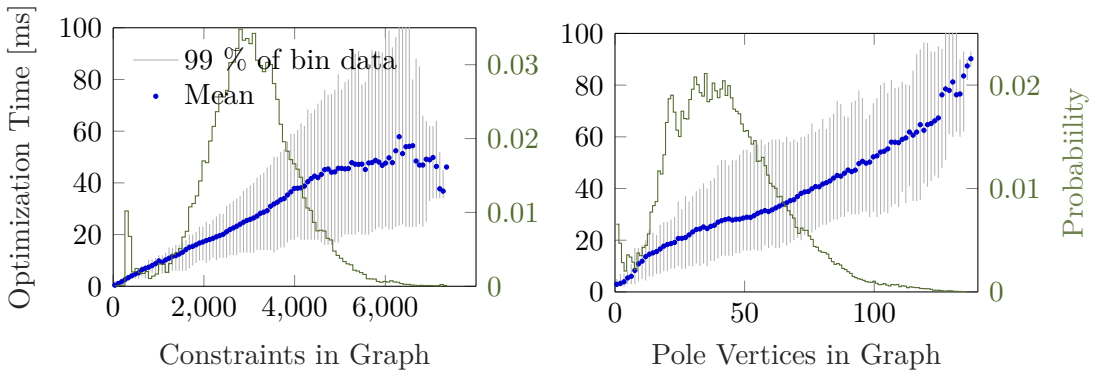


Figure 6.21: (a) Optimization Time and its dependency on the number of constraints in the graph. The number of constraints is the sum of odometry-, map-, polyline-, and pole measurement constraints in the sliding window graph of one algorithm cycle. In our case, the latter ones produce off-diagonal elements in the system matrix H, which increases the required optimization time. (b) Optimization Time and its dependency on the number of pole vertices in the graph. In our urban dataset, the number of pole vertices in the graph is equal to the number of map landmarks in the graph since we only include map matched landmarks in the graph. We discretized the x-axis of both plots into small bins, which we use for computing the y-axis mean values, shown in blue. The vertical gray lines denote the regions covering 99 % of the y values of each bin.

6.3 Particle filter vs. sliding window graphs

Following our argumentative comparison in Chapter 4.7, we experimentally compare the state estimation with particle filters against our graph-based sliding window approach in this section. We compare the key aspects of both approaches and support our claim that graph-based sliding window localization is favorable in terms of accuracy. Additionally, we evaluate the adaptive behavior of both approaches in terms of computational resources using the proportional-integral-derivative (PID) controller presented by Merfels and Stachniss (2017). Lastly, we demonstrate the beneficial capability of estimating old poses with our graph-based sliding window approach. The particle filter used in this comparison is described by Stess (2017) and briefly also by Wilbers et al. (2019b). In the scope of our experiments, the particle filter operates on the same input data and third-party map as our graph-based approach, which enables us to compare both approaches directly.

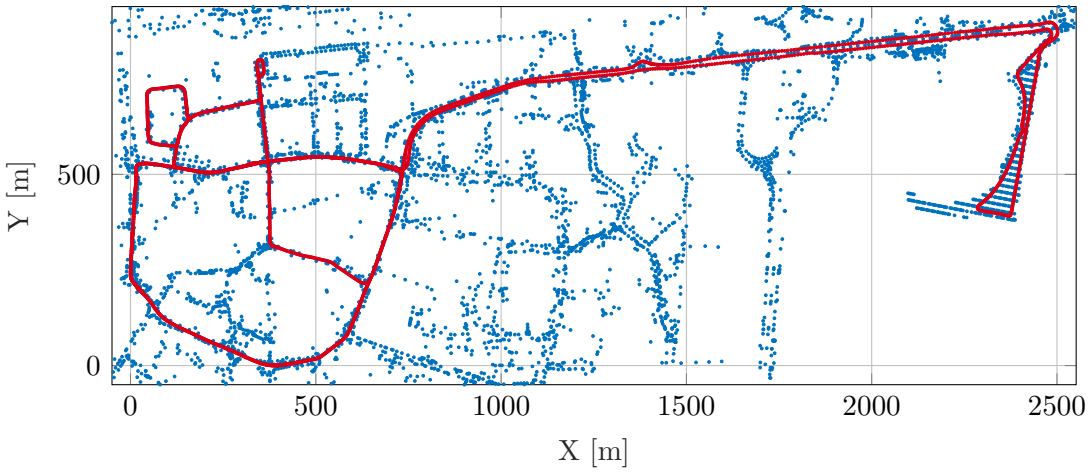


Figure 6.22: Map and trajectory of our real-world dataset recorded in Fallersleben. All pole-like landmarks in the test area are denoted in blue. The 16 km trajectory is denoted in red. It contains velocities between 0 km/h and 70 km/h and covers typical urban scenarios like heavy and light traffic, different street types, and different junction sizes.

The prototype vehicle used in this section is an earlier version of the e-Golf used in our Hamburg dataset with a similar sensor setup. A key difference is that the vehicle used in this experiment is equipped with Velodyne VLP-16 LiDAR sensors instead of VLP-32, which nearly halves the landmark detection range. Another difference is that we exclusively consider poles as landmarks and do not incorporate any other landmark types. The reference system used in this experiment is an Applanix LV 520 system without a wheel tick encoder, which, due to nearly perfect open sky conditions, operated flawlessly in its most precise positioning modes. The reference system is not used by our localization approach and only serves as a comparison. Our dataset for this experiment was recorded in

January 2018 in the district Fallersleben of Wolfsburg. The 16 km long trajectory and the third-party pole map is illustrated in Figure 6.22. We conduct a set of experiments in which we vary the size of the state vector. In the case of the particle filter, we directly control the state complexity of the estimation by setting the number of particles. In contrast, the state vector size of our graph-based localization is only partly influenced by controlling the number of poses. Additionally, we evaluate adaptively controlling the state vector size using the PID controller described by Merfels and Stachniss (2017). The target frequency of both approaches is set to 20 Hz. The poses inside of our sliding window graph are 20 ms apart. Within the experiments presented in this section, the lateral and longitudinal position variance of GNSS pose constraints included in the graph is set to a static value of $10\,\mathrm{m}^2$. We start by comparing the runtime behavior and accuracy of both approaches. Afterward, we demonstrate the accuracy benefit of estimating lagged poses with our approach.

6.3.1 Runtime behavior

This experiment is designed to show the relation between computation time and state complexity of the estimation. We compare the runtime behavior of a particle filter and our sliding window approach. In the case of particle filters, the complexity of the state estimation is controlled by the number of particles. In our case, we indirectly control the state vector size by setting the number of poses within a graph. In addition, we consider a PID controller variant that adaptively controls the state complexity of the estimation. Figure 6.23 demonstrates the results of our experiment. The figure shows that the PID controller manages to satisfy timing requirements by adjusting the number of particles and the number of poses in the graph. In both cases, the PID approach successfully provides pose estimates around our set target frequency of 20 Hz. In more detail, we show in Figure 6.24 that the required computation time depends on the number of constraints within the sliding window. Our insight is that the number of poses in a sliding window only implicitly influences the required computation time. More important than the number of poses is the number of constraints, which is influenced by the number of detected and map landmarks, as we have discussed in Section 6.2.5. The PID controller exploits this fact without the implicit knowledge of landmarks and successfully provides pose estimates with the set target frequency. Our results, illustrated in Figure 6.24, show that using a fixed sliding window with 500 poses is suitable for the above-described vehicle setup and a target frequency of 20 Hz. Likewise, Figure 6.23a shows that roughly 1000 particles are suitable for the particle filter when using a fixed number of particles. In sum, we compared the runtime behavior of both approaches and showed that controlling the sliding window size is a suitable option for limiting the runtime.

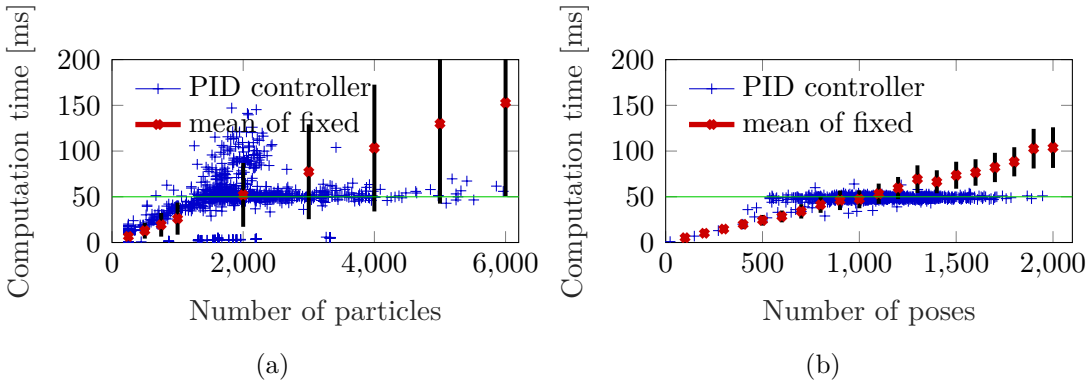


Figure 6.23: Relation between computation time and control variable. The green line in each plot marks the target output frequency of 20 Hz, which corresponds to the maximum computation time of 50 ms. The average computation time for each experiment is denoted as a red cross, with the single standard deviation in black lines. (a) For the particle filter, we control the number of particles. (b) For graph-based localization, we control the number of poses.

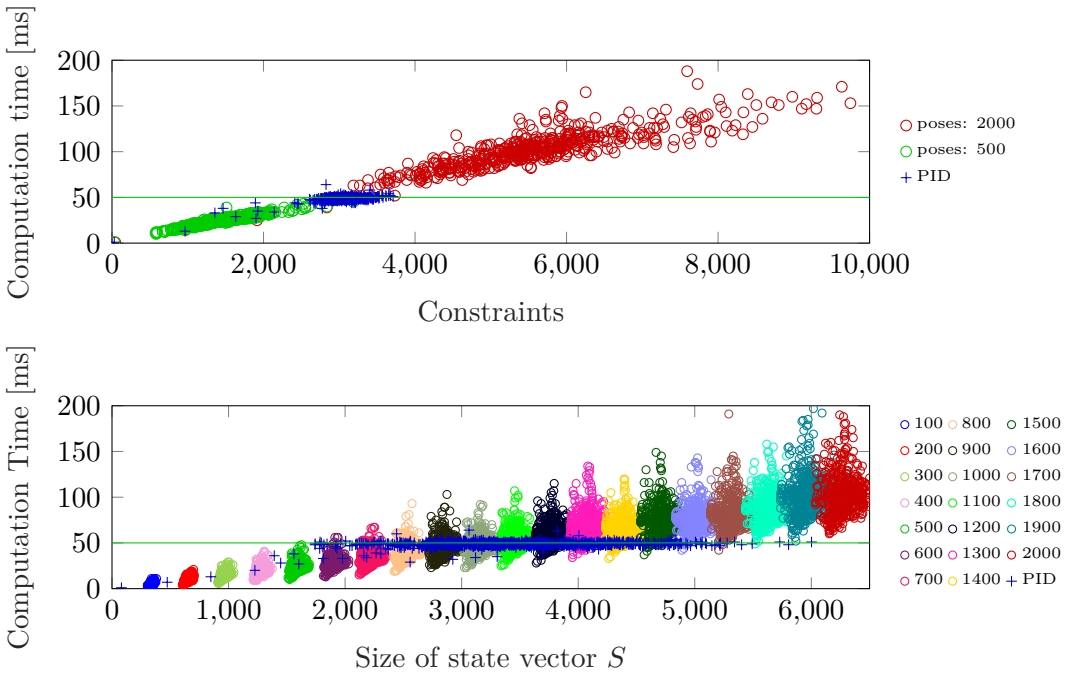


Figure 6.24: Influencing factors on the computation time of our graph-based sliding window approach in our Fallersleben dataset. The green line in both plots marks the target output frequency of 20 Hz, which corresponds to the maximum computation time of 50 ms. Top: Relation between computation time and the number of constraints included in a sliding window graph. The number of edges represents the number of measurements incorporated in the graph. Bottom: Relation between computation time and the size of the state vector $\boldsymbol{x}$ of a sliding window graph. In this experiment, the state vector's size is $S = 3N + 2M_p$, with N being the number of vehicle poses and M_p being the number of pole landmarks in the graph.

6.3.2 Accuracy

To compare the accuracy of our approach to localization to particle filters, we conducted a set of experiments with varying state complexities. In our experiments, both approaches operate on the same measurement data and map such that their only difference is the state estimation technique. Considering our above-mentioned vehicle setup and runtime analysis, a fixed number of 1000 particles and a sliding window with 500 poses are suitable values allowing online localization with 20 Hz. In this case, the achieved average Euclidean errors are 0.24 m for the particle filter and outperforming 0.17 m for our approach. We found that increasing the state complexity does not necessarily imply increased accuracy in both cases. The particle filter's accuracy starts to stagnate at roughly 1000 particles, while our sliding window approach does not benefit from more than 1000 poses in the sliding window. In the case of particle filters, increasing the number of particles corresponds to a more fine-grained approximation of the underlying probability distribution, which not necessarily means that the vehicle pose is estimated more accurately. In our case, increasing the number of sliding window poses means that our approach takes a longer time span of the past into account for estimating the current vehicle pose. In this sense, our interpretation is that the most recent past is more crucial for estimating the vehicle pose than the more distant past. Using the PID controller, which variably adjusts the state complexity depending on the runtime of previous algorithm cycles, improves the accuracy of our approach to 0.15 m using, on average, 1104 poses while still running at 20 Hz. In the particle filter case, the PID controller variant yields 0.24 m with, on average, 2160 particles. Our experiment highlights that the PID controller successfully exploits the available computation time, which can beneficially influence the accuracy under given runtime constraints. In both localization approaches, it is necessary to maintain at least a minimum number of states. Using only 250 particles yields an average Euclidean error of 0.27 m, whereas our graph-based approach yields 0.22 m with 100 poses. Both variants are less accurate than the experiments with the greater state complexity of the estimation. We present the discussed results of our experiment in more detail in Table 6.3. Furthermore, we compare our results based on their empirical cumulative distribution function (CDF) in Figure 6.25. The figure shows that our graph-based localization is superior to the particle filter in terms of accuracy as the CDF accumulates faster.

In sum, our experiment shows that our graph-based sliding window approach localization yields more accurate results and is thus favorable over particle filters for vehicle localization.

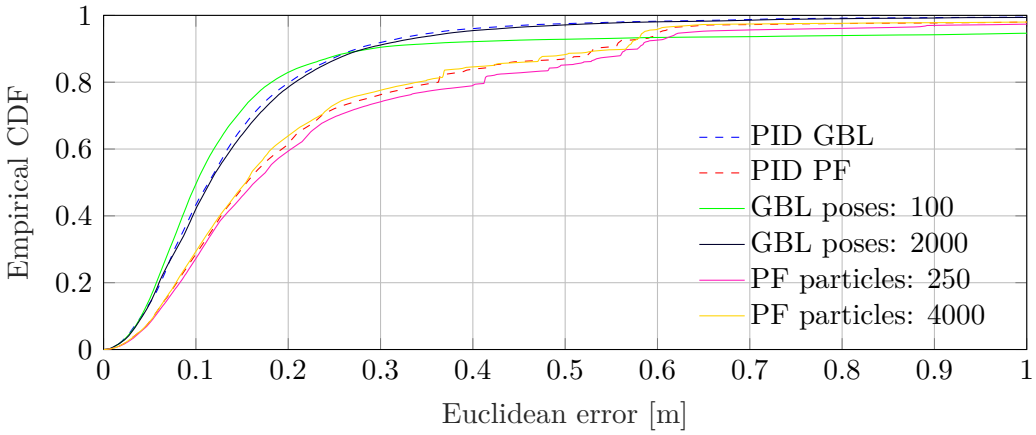


Figure 6.25: Empirical CDF for a set of experiments, comparing particle filter (PF) and graph-based localization (GBL). For clarity, we only illustrate an excerpt of our results. Table 6.3 provides further statistics on our results.

	State complexity	Euclidean	longitudinal	lateral	heading
GBL	100 poses	0.22 m	0.14 m	0.14 m	0.49 deg
	500 poses	0.17 m	0.12 m	0.10 m	0.39 deg
	1000 poses	0.15 m	0.11 m	0.08 m	0.34 deg
	2000 poses	0.15 m	0.11 m	0.08 m	0.34 deg
	PID (avg.: 1104)	0.15 m	0.11 m	0.08 m	0.34 deg
PF	250 particles	0.27 m	0.20 m	0.14 m	1.46 deg
	1000 particles	0.24 m	0.18 m	0.12 m	1.41 deg
	2000 particles	0.24 m	0.18 m	0.10 m	1.38 deg
	4000 particles	0.24 m	0.18 m	0.11 m	1.38 deg
	PID (avg.: 2160)	0.24 m	0.18 m	0.12 m	1.42 deg

Table 6.3: Average Euclidean, longitudinal, lateral, and heading errors for different configurations of a particle filter and our graph-based approach during a 16 km urban drive in Fallersleben. The used prototype vehicle is equipped with Velodyne VLP-16 LiDAR sensors. For the PID experiments, we denote the average number of particles and poses in the graph in brackets.

6.3.3 Estimating past poses

This experiment demonstrates the benefit of optimizing past poses in our graph-based approach. In contrast to particle filters, re-estimating the trajectory within the sliding window is part of every algorithm cycle within our graph-based optimization approach. Consequently, over time the estimates of past vehicles poses are affected by more recent measurements, which improves their accuracy. Figure 6.26 shows box plots for the Euclidean errors within the trajectory of a sliding window graph. The accuracy in the middle of the graph is better than at the head and tail. The reason for this effect is that the poses in the middle of the graph are more constrained than the poses at the ends of the graph. The effect at the tail of the graph is caused by using truncation instead of marginalization. In sum, our experiment highlights that estimated poses in the past are more accurate than the most recent vehicle pose. While automated driving applications usually require the most recent pose estimate, using more accurate but lagged poses can be beneficial for non-timing-critical applications.

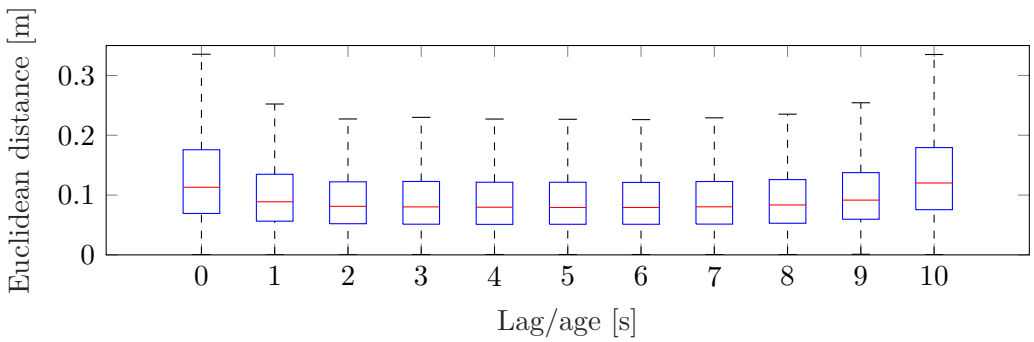


Figure 6.26: Accuracy within the trajectory of a graph. A lag of 0 s represents the most recent pose at the head of the graph, whereas a lag of 10 s denotes the oldest pose at the tail of the graph. This plot corresponds to a graph with 500 poses that are 20 ms apart and is based on our Fallersleben dataset. The figure shows that the most accurate poses are in the middle of the graph.

6.4 Implications of including low-cost GNSS

This experiment is designed to show the effect of including the pose estimates of a standard-consumer low-cost GNSS into our graph-based localization. Additionally, we demonstrate the capability of our approach to work in GNSS-denied regions, for which we only use GNSS once during initialization in one of our experiments. This corresponds to a complete GNSS outage after the initial startup. We compare in Table 6.4 the performance in terms of mean Euclidean error, average lateral error, and average longitudinal error of three different system variations.

Our experiment is based on our Fallersleben dataset, which does not contain any GNSS outage, which would distort our results. To compute the errors, we register the ground truth trajectory of our reference system to the timestamps of the pose estimates and afterward compute the mean Euclidean distance vector for each estimate. Our experiments show that including GNSS measurements worsens the accuracy of our graph-based sliding window approach. This effect is less distinct the higher the GNSS variances are. In relation to graph-based optimization, a higher GNSS variance can be interpreted as a down-weighting of the GNSS data compared to the other inputs. Given that our approach works best without GNSS at all, this explains our observation. We attribute the negative effect of including GNSS pose estimates to the Kalman-filter based preprocessing inside the GNSS-receiver. Usually, this preprocessing handles multipath effects insufficiently. As a result, the GNSS-based estimates are significantly biased. Naively incorporating this biased data into our sliding window graph results in biased pose estimates and, therefore, less accuracy.

Although we showed that our system is able to operate without GNSS over an extended period of time, we prefer in practice to still include it. Despite the negative effect on accuracy, GNSS is still useful in scenarios without a map at all and provides a valuable option for recovery. The issue of including biased data into graph-based optimization approaches is discussed in more detail by Noack et al. (2015). Another method that corrects biased GNSS data in a preprocessing step is presented by Merfels et al. (2016). An approach that directly incorporates GNSS pseudoranges, instead of already preprocessed pose estimates, into graph-based optimization has been presented by Sünderhauf (2012). While all of these methods, would potentially alleviate the bias issue, we found that simply scaling up the variance to a high value already is sufficient for our use case. As we have discussed, this down-weights the influence of the GNSS pose estimates on the optimization result. Beneficially, the constraints are still included in the graph such that it is still globally constrained and corrected in areas without any map data but practically ignored if other map data is included in the graph. Comparing the accuracy results of our Hamburg dataset with an average mean error of 0.11 m, in which we chose $\sigma = [1000, 1000, 1000]$, with the 0.10 m error achieved with our approach without GNSS (see Table 6.4) suggests that simply scaling up the GNSS variance is a viable option that alleviates the negative impact on accuracy.

In sum, our experiment highlights that it is crucial to consider the potential bias inherent in GNSS pose estimates when including them in our graph-based sliding window approach. We showed that scaling the covariance is a suitable option in practice and emphasized that our approach also works if GNSS is only used once for global initialization.

error type	GNSS	GBL + GNSS (σ_1)	GBL + GNSS (σ_2)	GBL + GNSS init
lateral	0.95 m	0.15 m	0.10 m	0.08 m
longitudinal	1.00 m	0.12 m	0.12 m	0.06 m
heading	5.54 deg	0.42 deg	0.39 deg	0.25 deg
Euclidean	1.53 m	0.22 m	0.17 m	0.10 m

Table 6.4: Absolute mean errors in our 16 km long urban drive in Fallersleben. The table shows the errors of our graph-based localization approach in three variants. We choose the longitudinal, lateral, and heading static variances for all GNSS constraints as $\sigma_1 = [2, 1.5, 6]$, comparable to the receivers specification, for our first variant. In our second variant we set $\sigma_2 = [100, 100, 600]$. Our third variant (GBL) only uses GNSS once for initialization and afterward completely neglects GNSS information for the whole test drive, which yields the lowest errors across all variants.

6.5 Impact of our data association strategy

This experiment demonstrates the ability of our system to revise associations between detected and map landmarks. We explained our concept for revising associations in Chapter 4.4.7, where we presented an example of such a revision in Figure 4.22. During our 16 km Fallersleben test drive, the map associations were revised by our system 35 times. In an additional experiment, the ability to revise associations was turned off. The mean Euclidean error increased from 10.3 cm to 20.1 cm. This highlights the need for subsequent verification of previous map associations. Therefore, our strategy for revising associations inside the graph construction based on subsequent additional information is beneficial for our system's performance.

6.6 Delayed map refinements

The main focus of the following experiments is the ability of our localization approach to propose map refinements. The term *delayed map refinements* emphasizes that we do not directly apply the map refinements but first transmit them to a back-end service for validation, as we have presented in Chapter 5. This includes adding new landmarks and changing existing ones. Since the back-end validation is not part of this thesis, we focus our evaluation on the map refinements computed within the vehicle. We begin by evaluating our sparsification scheme for marginalization in sliding window graphs, which we presented in Chapter 5.3. It is closely related to the ability of our approach to add previously unknown landmarks to a map. This part of the evaluation reflects our previously reported results (Wilbers et al., 2019c). As an addition, we demonstrate the capability of our approach to refine existing map landmarks in Section 6.6.5.

Our evaluation of sparse global priors is based on simulated data, which allows

us to use error-free ground truth data as reference, and real-world data collected with our e-Golf prototype vehicle. The datasets are illustrated in Figure 6.28 and Figure 6.27. In the scope of this evaluation, we only consider pole-like landmarks. If not mentioned otherwise, we randomly delete 20% of the map landmarks in our datasets, which are then re-estimated with our approach. These re-estimations are the basis for our evaluation. In the context of our map refinement architecture, the estimated landmarks are additions to the map. We support our claim that our approach is able to improve existing map landmarks in a separate experiment in Section 6.6.5.

6.6.1 Approximating marginalization with sparse global priors

The first experiment on delayed map refinements is designed to evaluate if we are able to approximate marginalization with our *sparse global prior distribution*. Therefore, we calculate the Euclidean distance between the estimated landmark positions that we add to the map and the corresponding full graph solution based on simulated data. We compare our sparse approach Equation (5.11) to dense marginalization as in Equation (5.6), and no marginalization at all. We use the latter as a baseline and normalize all errors with its mean. In Section 6.6.3, we investigate the covariance estimates separately and here only consider the estimated mean positions. Table 6.5 highlights the benefits of dense marginalization with a local linearization point, as described by Eckenhoff et al. (2016), with 53.4% of the errors compared to no marginalization. Its approximation with our sparse global prior approach performs with 48.2% even slightly better. The sparse variant performs better than the dense one consistently in all experiments shown in Table 6.5 regardless of the chosen linearization point. Besides numerical errors, we explain this positive effect by linearization errors, which for our sliding window setting have a stronger impact in the dense case. Our approach is explicitly designed to neglect dependencies between states. In contrast, in the dense case, inaccurate dependencies induced by marginalization accumulate. The comparable results show that we are successfully able to approximate marginalization with sparse global priors. The advantage of our approach over dense marginalization is that it prevents fill-in in the system matrix H, which would negatively influence the required computation time during optimization. Moreover, our approach of decomposing dense marginalization into sparse priors has the advantage that it eases the data handling within the sliding window graph. Each sparse prior can not only be individually inspected but also individually robustified by applying robust cost functions. Overall, our experiment supports our claim that our sparse global prior approach approximates dense marginalization. We showed that our

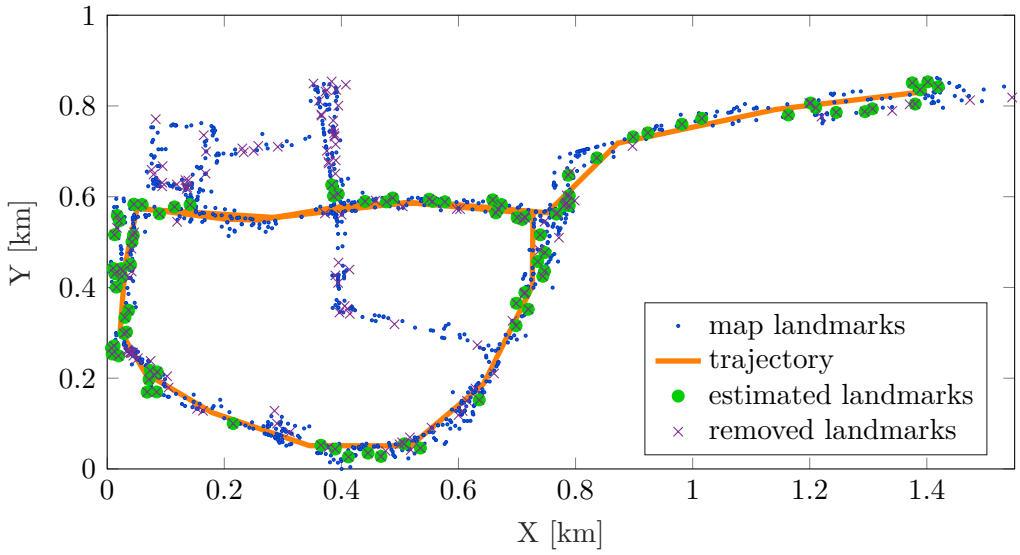


Figure 6.27: Map and trajectory of our real-world dataset. We randomly remove 20% of the landmarks from the map to evaluate our method. The map was recorded several months before the actual test drive. The trajectory and pole-like landmark measurements were recorded with our e-Golf prototype vehicle. The figure illustrates the estimated landmarks relevant to our evaluation in green. These are a subset of the removed landmarks.

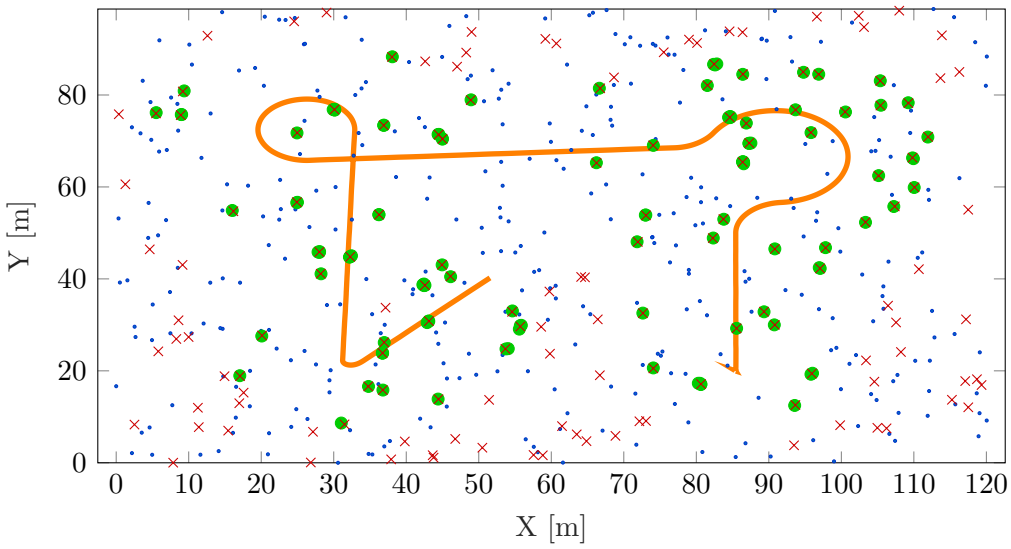


Figure 6.28: Map and trajectory of our simulated dataset. All blue landmarks are used for localization. The red crosses represent the ground truth for the estimated landmark positions in green. The notation is similar to Figure 6.27. The figure provides an impression of our simulated dataset.

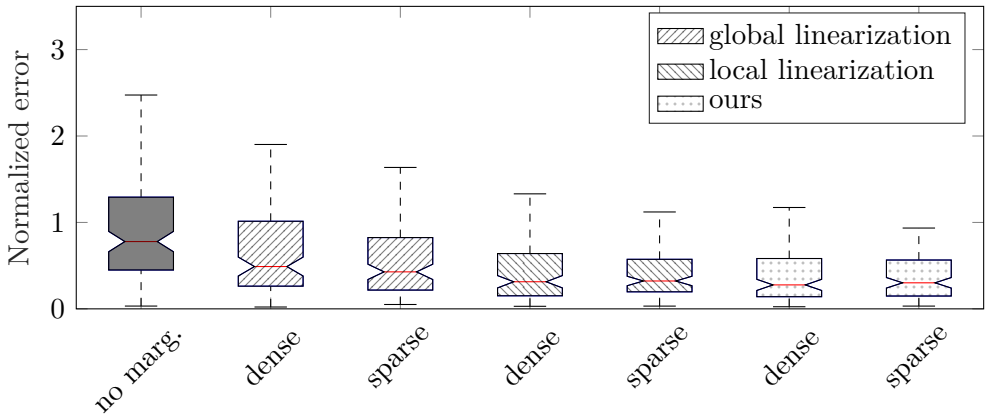


Figure 6.29: Boxplots of the normalized Euclidean distances between sliding window estimates compared to the full graph solution based on our simulated dataset.

	global linearization	local linearization	ours
dense	73.5 %	53.4 %	46.7 %
sparse	61.1 %	50.9 %	48.2 %

Table 6.5: Normalized average errors expressed as percentages. The baseline is the average error without marginalization (100%), such that lower is better in this table. The results are based on our simulated dataset.

approach yields comparable results without inducing fill-in in the system matrix H during optimization.

6.6.2 Global vs. local linearization vs. our approach

Our second experiment on map refinements supports the claim that our approach of utilizing global linearization points yields similar results compared to using local linearization, which requires optimizing the marginalization blanket. We compare the use of global and local linearization points to our method of utilizing global linearization points and including a gradient term, as shown in Equation (5.10). The comparison for the dense and sparse case is shown in Figure 6.29. It can be seen that using naive global linearization points still performs better than no marginalization but yields the worst performance of all marginalization approaches. The figure shows that our approach performs even slightly better than local linearization in the sparse case. We contribute this effect to our sparsification scheme. We first approximate marginalization with a sparse distribution and afterward include a gradient term in our distribution and not the other way around. By doing so, we avoid the effect of inaccurate dependencies induced by marginalization between states on the gradient. Additionally, our experiment suggests that even in the dense case, it is beneficial to use the global linearization point and consider the gradient term, as we suggest in Section 5.3.5, rather than

using the local linearization point. Although this might not generalize to arbitrary graphs, it is beneficial for our sliding window case. Overall, the comparison shows that our approach provides comparable results to local linearization and successfully utilizes global linearization points. Compared to local linearization, our approach does not require an additional optimization step and is therefore preferable.

6.6.3 Conservative estimates

The third experiment in this sequence supports our claim that our approach estimates conservative landmark positions. Based on our simulated dataset we calculate the Mahalanobis distance between the estimated landmark with its covariance and the artificially removed landmark position from the map. Figure 6.30 shows the result for the different marginalization and sparsification variants. If the given percentile levels from a specific experiment are above the given percentile boundaries, the covariances are overconfident (too small). Vice versa, if the levels are below the boundaries, the variances are underconfident (too large). In our case, it is favorable to underestimate the covariances to be more robust against outliers. The figure shows that simply using the global linearization point without considering the gradient term, as described in Section 5.3.5, is suboptimal and produces overconfident results. In comparison, local linearization and our approach produce underconfident results. This is preferable within our use case of estimating landmark positions as map refinements for back-end validation.

6.6.4 Sparsity pattern and accuracy of landmark additions

Finally, we show that our approach has exactly the same sparsity pattern compared to no marginalization but provides more accurate estimates. The results of this experiment are based on the Euclidean distance between the estimated landmark positions and the deleted ones from the map. Figure 6.31 compares our approach to dense marginalization and shows that our system matrix H achieves a better sparsity pattern. It is by design exactly similar to using no marginalization. Compared to dense marginalization, our approach has the advantage that the sparsity pattern of the system matrix H does not suffer from fill-in over time. Therefore, the system matrix H remains sparse, which is beneficial for the required computation time during optimization. Together with Figure 6.32, which shows that our approach achieves better accuracy than using no marginalization and is comparable to dense marginalization, the experiment supports our claim.

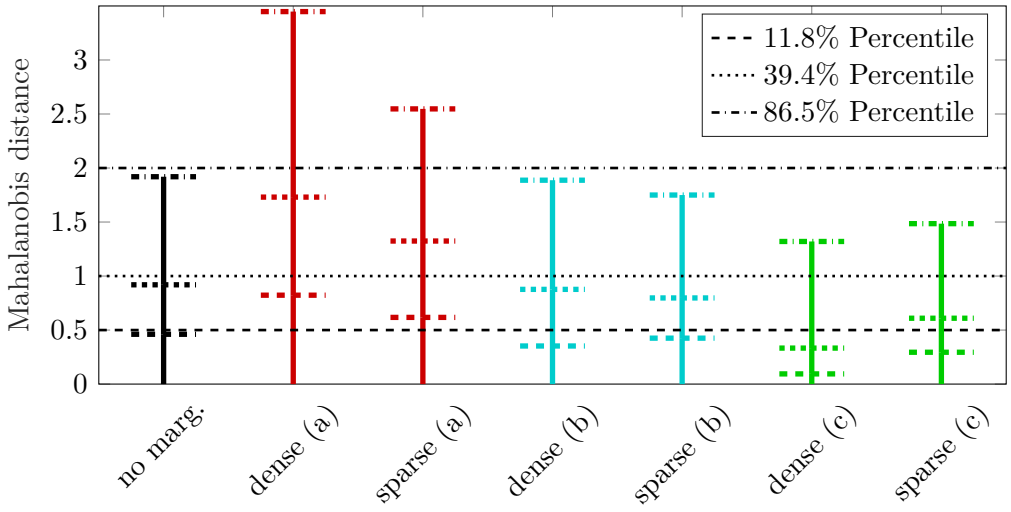


Figure 6.30: The figure shows that the estimated covariances fit the errors between estimated landmark positions and artificially removed landmarks of our map based on our simulated dataset. The plot compares (a) global linearization, (b) local linearization, (c) our approach. The figure shows that global linearization produces overconfident results, while local linearization and our approach yield underconfident, i.e., conservative, estimates. The latter is preferable for our map refinement use case. The results shown in this figure relate to the normalized Euclidean distances shown in Figure 6.29 by illustrating how reliable the corresponding covariance estimates are.

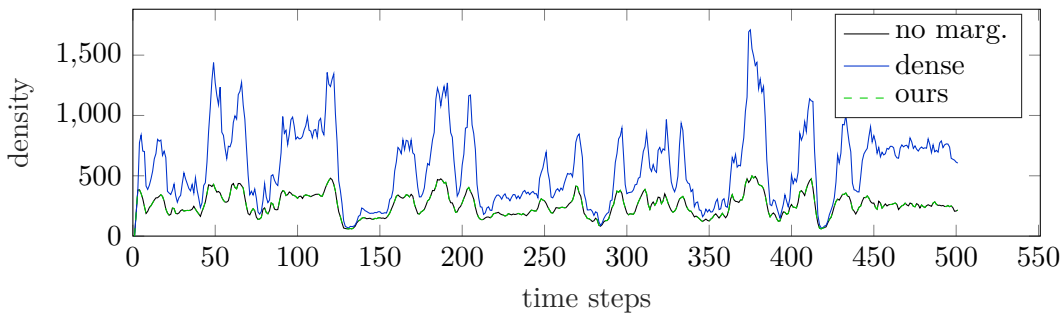


Figure 6.31: The figure shows the sparsity patterns of dense marginalization, no marginalization, and our approach, based on our real-world dataset. The density value shown in the plot is the number of non zero block-diagonal entries in H for each time step. Our sparse approximation has the exact same sparsity pattern as without marginalization.

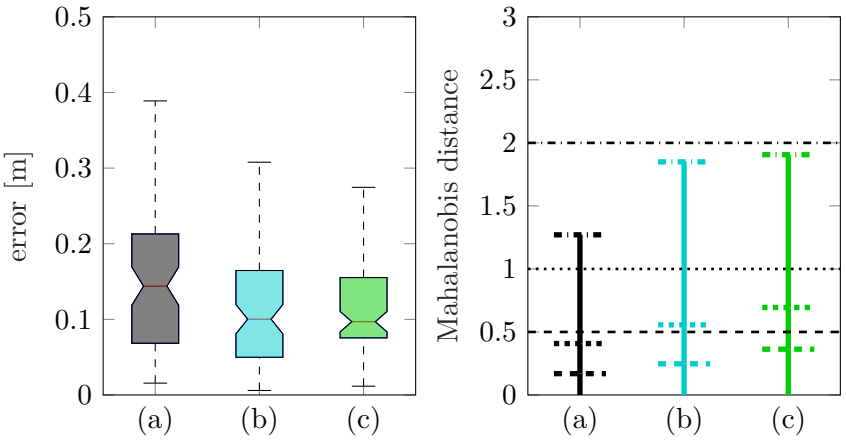


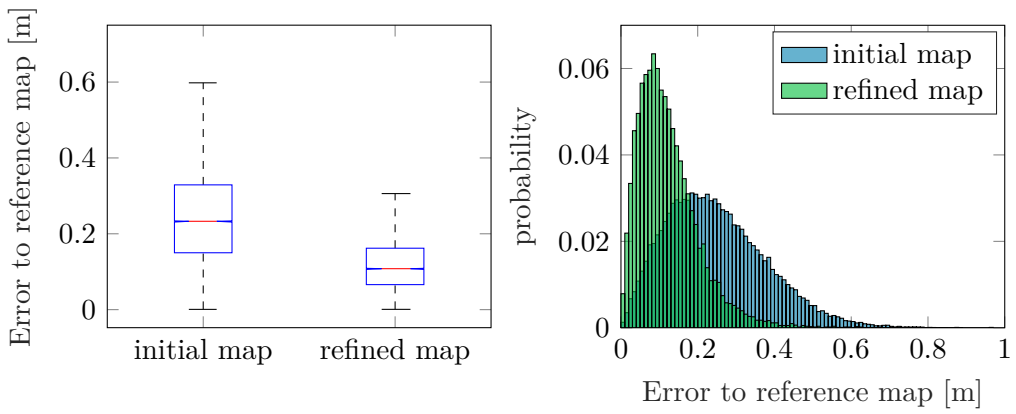
Figure 6.32: Results for our real-world data set. The plots compare (a) no marginalization, (b) dense marginalization with local linearization points, and (c) our method with sparse global priors. The plots show that our approach provides similar performance to dense marginalization. Considering that our approach has a favorable sparsity pattern, it is clear that our approach is superior to standard dense marginalization.

6.6.5 Modifying map landmarks

In this experiment, we demonstrate the capability of our sliding window approach to refine existing map landmarks. Instead of directly including the computed refinements into our map, we store the estimated landmark positions for post-processing as described in Chapter 5. Here, we investigate the accuracy of the estimated landmark positions based on a dataset, which we illustrate in Figure 6.34 and explain in the following.

For the purpose of this experiment, we add artificial noise to the position of each individual landmark stored in the map. We choose to add independent Gaussian noise $\mathcal{N}(\mu = 0, \sigma = 0.2)$ to the *Easting* and *Northing* coordinate of each landmark. Figure 6.34a illustrates the noisy map with its error to the reference map. We use the artificially noisy map as the input map for our graph-based sliding window localization. Our approach refines the landmark positions during localization, which we store and compare to the original reference position before adding noise. To ensure that errors in the original map do not distort the results, we perform this experiment based on partly simulated data. Therefore, we take a real trajectory from one of our drives in our Hamburg dataset and sample landmark measurements with Gaussian noise $\mathcal{N}(\mu = 0, \sigma = 0.1)$ around the true landmark position based on the vehicle pose and the original reference map. To model more realistic measurements, we set the probability that a landmark is measured at a specific timestamp to 10 % and the maximum detection range to 50 m. Our experiment shows that our approach improves the noisy map with an average Euclidean error of 24.7 cm to a refined version that only has an average Euclidean error of 12.3 cm. In more detail, Figure 6.33 shows the boxplots and

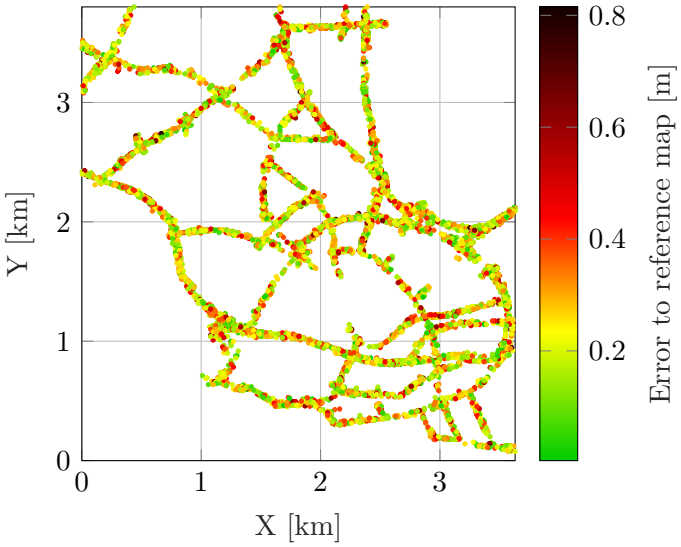
histograms for the error distributions of the initial noisy map and the refined map. Moreover, Figure 6.34b shows the refined map landmarks along the driven trajectory colored by the remaining error to the reference map. In summary, the experiment supports our claim that our approach is able to improve existing map landmarks.



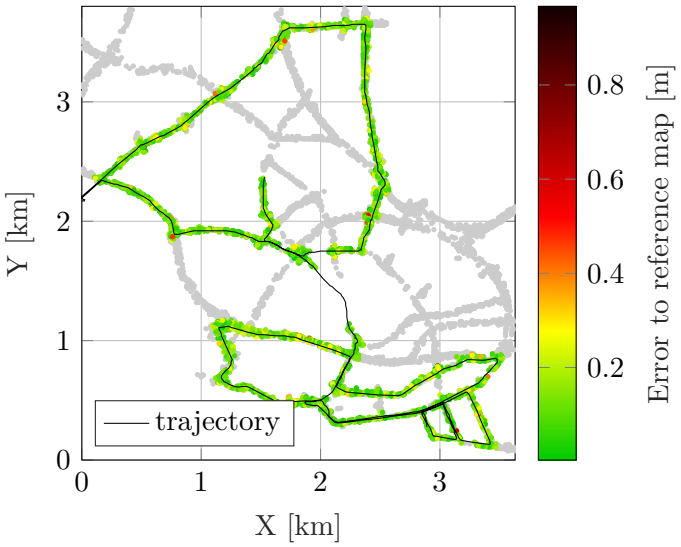
(a) Boxplots

(b) Histograms

Figure 6.33: Error statistics of refined map landmark positions. The figures show that our sliding window map refinement approach successfully improves the accuracy of an initial noisy map from an average Euclidean error of 24.7 cm to 12.3 cm.



(a) artifically noisy map



(b) refined map after one pass

Figure 6.34: Results for our experimental evaluation to demonstrate the capability of our approach to refine existing map landmarks. We here illustrate the map landmarks colored with the error to the reference map and provide the statistics of our experiment separately in Figure 6.33. (a) Artificially noisy map that we produced by adding noise to the reference map. We use the shown map in this experiment as an input to our graph-based sliding window approach. (b) The figure illustrates the refined map landmarks along the driven trajectory colored with the error to the reference map. Our approach clearly improves the map. Landmarks outside the detection range remain unchanged and are colored in gray.

6.7 Summary of the evaluations

In this chapter, we experimentally evaluated our graph-based localization approach with respect to pure localization and map refinement. We started by introducing our urban dataset recorded in Hamburg covering 319 km driven distance and provided an impression of the wide variety of included sceneries. To set our results into perspective, we presented a detailed analysis of our wheel-tick and IMU-based odometry module and compared it to a visual odometry approach. We showed that assuming a near drift-free odometry during local association is a valid assumption of our sliding window approach that holds in practice. Our results suggest that our approach is transferable to sensor setups relying on visual odometry instead of wheel-tick and IMU sensors. Afterward, we investigated the localization performance of our approach w.r.t. urban driving using a general-purpose third-party map. Yielding an average Euclidean error of 0.11 m and an availability of 99.97 % w.r.t. a 0.5 m average Euclidean error boundary, our experiments support our claim that our presented approach is applicable for urban automated driving. Compared to GNSS-based systems, our approach does not suffer from GNSS outages in, e.g., urban canyons and reliably works in urban scenarios. In relation to computational tractability, we provided a detailed runtime analysis of the different steps in our algorithm. Our approach requires, on average, 54 ms for an algorithm cycle and thus is applicable for online localization. Furthermore, we demonstrated that sliding window graphs are favorable over particle filters in terms of accuracy. We compared their runtime behavior and demonstrated the accuracy benefits of estimating past poses. Additionally, we experimentally investigated the effect of including pose estimates in our graph based on a low-cost GNSS receiver and found that including biased GNSS data negatively impacts the accuracy of our approach. We demonstrated that scaling the covariances of GNSS constraints is viable in practice for mitigating the negative effects and showed that our approach even works best if GNSS is only used once for global initialization. We separately conducted a set of experiments evaluating our map refinement approach. Our evaluation supports our claim that our sparse global prior approach approximates dense marginalization while using global linearization points. We showed that our approach maintains the same sparsity pattern as without marginalization but at the same time improves the accuracy of estimated landmark positions. Besides, we demonstrated that our approach contributes to improving the accuracy of landmarks in existing maps.

In sum, our experimental evaluation supports our claim that our presented graph-based sliding window approach using third-party maps is suitable for vehicle localization and map refinement in urban areas and therefore enables automated driving applications.

Chapter 7

Conclusion

Automated driving functions are one of the key technologies in increasing road safety and mobility. In this thesis, we presented an effective and applicable approach for localization of automated vehicles. Knowing the precise position and orientation of a vehicle within a map at any point in time allows enriching perception, scene understanding, and planning tasks with map data. The map data contains information about the vehicle's environment, which the vehicle might not be able to sense with its sensors. For example, large trucks surrounding our vehicle or challenging weather conditions might restrict the visibility of traffic signs, such that the vehicle needs to rely on localization to infer the traffic rules from the map.

Nowadays, satellite-based localization techniques, like GPS, have been improved up to centimeter-level accuracy by incorporating real-time error correction data. These approaches work well under open sky conditions with direct line-of-sight satellite visibility and are commonly used in several applications where this constraint holds or when humans supervise the automated system. A major disadvantage is that autonomous systems suffer in situations with limited satellite reception, which especially occur in urban environments.

In this thesis, we presented a localization framework that relies on fusing landmark-, odometry-, GNSS-, and map data for estimating highly precise vehicle poses and map refinements. Compared to GNSS-based localization, our approach provides accurate and reliable pose estimates, even in challenging urban environments with limited satellite reception. In fact, our approach only requires GNSS for initialization and does not need it otherwise. We investigated the challenge of incorporating general-purpose third-party map data, which we consider an important aspect for scaling autonomous driving applications to mass markets. Our maps contain static 2D landmarks like poles, building facades, and road markings, which consume much less memory than, e.g., dense LiDAR maps. We extended our approach to compute landmark map refinements within the vehicle

to transmit them to a back-end service. Compared to transmitting raw data, our approach conceptually requires much fewer data to be transmitted, which is beneficial for over-the-air updates.

We extensively tested our localization approach under real-world conditions, where we, among others, used trajectory data computed on-board within a prototype vehicle for our evaluation. Our experiments consider real-world challenges like extended GNSS outages, stop-and-go traffic, heavy and light traffic, as well as various environmental conditions. We showed that our data association approach is applicable for incorporating landmarks from various sensors and can be used with general-purpose third-party maps. Also, we provided evidence that our map refinement approach is applicable for generating accurate and conservative map update hypothesis. Overall, our experiments suggest that the proposed approach is applicable to urban automated driving. It is currently used in several automated prototype vehicles, including trucks and the presented e-Golf model.

7.1 Short summary of key contributions

Our main contribution presented in this thesis is the design and realization of a graph-based sliding window localization approach for automated driving applications. We investigated five different aspects in this work.

First, we proposed a localization architecture for graph-based optimization that splits data association into the parts *local association*, *map matching*, and *temporal association smoothing*. In brief, we first locally associate individual landmark detections to each other to identify measurements that belong to the same landmark. Afterward, we compute the best transformation between the local map that contains all identified landmarks and the global third-party map. While we compute the first two steps based on the data within the current sliding window of each algorithm cycle, we aggregate the found map matches over all algorithm cycles to infer the overall best matches for our optimization graph. All parts combined allow us to revise map associations and perform delayed association such that the detected landmarks are reliably matched to the map. We include the identified map associations, landmark-, odometry-, and GNSS measurements in a sliding window graph over vehicle poses to limit the state vector size and the number of measurements that we consider during optimization. In return, our approach is computationally tractable for online localization. We analyzed the properties of our approach and provided an in-depth evaluation.

Second, we contributed to utilizing general-purpose landmark maps for localization. These are typically created by a third-party distributor and not tailored towards a specific sensor setup. In return, the landmarks stored in the map and the landmarks that a vehicle can detect may only partially overlay, which

we especially take into account in our data association framework. We showed how to derive and integrate the individual map landmarks as priors in our factor graph approach. Our approach is beneficial for using landmark measurements from various sensors for localization on a single general-purpose landmark map. It reduces the effort for deploying our approach on vehicles with different sensors that still use the same general-purpose map.

Third, we contributed to graph-based sliding window optimization by introducing a novel sparsification scheme that limits the information loss when removing measurements from the sliding window. We derived novel *sparse global prior* that approximate dense marginalization without the drawback of inducing a fill-in in the optimization. Our approach approximates the removed information in individual priors for states such that it has the exact same sparsity pattern as the problem without marginalization. In addition, we proposed how to utilize global linearization points instead of local ones.

Fourth, we presented a landmark map refinement scheme that uses our localization architecture and sparse global prior approach. We showed how our approach contributes to estimating accurate and conservative landmark positions for previously unmapped and changed landmarks. Thereby, our focus is on computing the landmark positions as update hypothesis that we transmit to a back-end server.

Our fifth contribution is the comparison between including 2D point and polyline-based landmarks in graph-based optimization techniques. We derived the error functions and factors for both landmark types and discussed their influence on having a full rank equation system during graph optimization.

In sum, our graph-based sliding window approach contributes to vehicle localization for automated driving applications. In sum, our contributions yield a graph-based sliding window approach for localization that is applicable for automated driving. Our approach provides highly accurate global pose estimates even in challenging urban environments and is computationally tractable such that it is fast and frequent enough for online localization. Moreover, our approach incorporates landmarks from multiple sensors, works with general-purpose third-party landmark maps, and contributes to map refinement for reliable long-term localization. Overall, our contributions lift the potential of graph-based localization towards enabling autonomous driving applications on a large scale.

7.2 Limitations, outlook, and discussion

In this thesis, we presented our localization framework that incorporates landmark measurements in a generic way. While we consider this a substantial advantage for developing a flexible and transferable approach, it also comes with the compromise of ignoring the error characteristics of underlying sensors. We require

that all landmark measurements are given in the vehicle reference frame (VRF) and treat all landmark measurements in the same way. Concerning the error characteristics, we see two improvements that could be made within our graph-based localization approach. On the one side, graph constraints from different sensor sources could be treated with different error and robust kernel functions, which are customized depending on their natural error characteristics. On the other side, incorporating sensor measurements directly in their natural coordinate system is a promising room for improvement. For example, it is a challenge for front-facing cameras to provide highly accurate depth estimation for objects such that the transformation to VRF is error-prone. In this case, a tighter coupling with error functions directly in the sensor reference frame is a promising way of improving the impact of cameras in our system. Nevertheless, both suggestions trade the flexibility of our system for, yet to be investigated, effects on accuracy and robustness. Overall, whether a tighter sensor integration is required depends on the use case and the vehicle's sensor setup's overall capabilities.

Another limitation of our current approach is the independent and identically distributed Gaussian noise assumption for all incorporated measurements. While this is common in literature, it often does not hold in practice. Especially landmark measurements that are heuristically computed within a black box detector are not only likely to follow a different distribution, but subsequent measurements might even depend on each other if the detector internally applies tracking. As a consequence, the measurement fusion within our optimization is suboptimal. Considering that optimal measurement fusion is a general challenge across various domains, multiple approaches have already been developed that might adapt to our use case. Given that the achieved accuracy of our presented approach is already sufficient for automated driving, we expect that, at least in our use case, the expected margin of improvement might be negligible. Nevertheless, with other sensor setups, a more distinct noise modeling might be more crucial.

Within this thesis, we limited ourselves to computing 2D poses. Although this is adequate for many automated driving applications, some require a 3D pose with height, roll, and pitch estimation. For some of these use cases, it is sufficient to directly complement the 2D pose with the missing 3D information from, e.g., IMU and GNSS readings. Since computing 3D poses is standard in closely related SLAM approaches, adjusting our graph-based approach to estimating 3D poses should be straightforward. This could be done in several ways, one of which would be only to adjust poses to be in 3D while continuing to use 2D landmarks. Besides extending the state vector, the main adaptation would be to extend the involved error functions such that 3D measurements are properly integrated. In all cases, we consider it important that the optimization problem remains fully constrained such that no further assumptions need to be made dur-

ing optimization. Compared to estimating 3D poses, our main advantage of only considering 2D poses is the reduced computational effort, which is a crucial point to take into account.

A promising concept that would enhance our approach is to consider more distinct semantic landmark types. In this thesis, we distinguished between landmark measurements in a very generic way, mainly based on their physical geometry. For example, we used cylindrical objects like lamp posts, tree stems, and reflector posts and jointly considered them as poles in our approach. Considering sensors, like LiDAR and radar, it is nowadays still a hard task to further classify detected geometries into semantic objects only based on the sensor data. In the scope of this thesis, we considered the provided landmark detectors as given and did not further investigate improving them. Improvements in combining camera-based object classification with, e.g., LiDAR point cloud data could ease inferring more distinct landmark types with enhanced semantics. The more reliable semantic details are available, the easier is local association and map matching. We deem it possible that incorporating more reliable semantic information is a major advantage for localization. Likewise, it would also improve map refinement as landmarks would get more distinguishable on the back-end server.

Another aspect that could improve landmark and map-based localization approaches is to target an optimal detector and map overlap, i.e., the map only contains elements that the vehicle can detect and detectors have a near-zero false positive rate. In this thesis, we explicitly investigated using general-purpose maps that are not tailored towards a specific sensor setup. However, long-term map refinements could tailor the map content over time. It is an open question if a back-end fusion of fleet data maintains the general-purpose characteristics of a map. While this certainly depends on the size and variety of the fleet data, it would also be interesting to investigate the challenges in automatically obtaining subsets of a single general-purpose map that are tuned for specific sensor setups. In relation, incorporating more semantic and distinct landmark types is a beneficial factor for generating tailored maps.

Overall, our main conclusion is that the key concepts presented in this thesis contribute to providing an accurate and reliable localization system for automated driving. Our graph-based sliding window localization approach was used in various projects and proved to be useful in many different scenarios. In this thesis, we tackled incorporating landmarks in a generic way such that our architecture easily integrates landmark measurements from different sensors. While having a generic system is especially useful in the research and development context, a more tailored integration might be required for series development. In this sense, we favor incorporating landmarks in a generic way whenever possible but likewise think that a deeper and more sensor-specific coupling is favorable

in other cases (e.g., no LiDAR, nor radar). We investigated using third-party landmark maps for localization and showed that using general-purpose maps is a technically feasible option. We think that our approach could be used for scaling landmark-based localization to mass-market applications. Lastly, we pointed out multiple aspects for further research and discussed improvements that would help to lift our presented approach out of the research context towards series production.

Acronyms

ADAS advanced driver assistance system

CDF cumulative distribution function

DCS dynamic covariance scaling

DGPS Differential GPS

EgM EgoMaster

EKF Extended Kalman Filter

FN false negative

FP false positive

GLONASS Globalnaya Navigazionnaya Sputnikovaya Sistema, or Global Navigation Satellite System

GNSS Global Navigation Satellite System

GPS Global Positioning System

ICP iterative closest point

IMU Inertial Measurement Unit

INS inertial navigation system

IRLS iterative reweighted least squares

LiDAR light detection and ranging

LM Levenberg–Marquardt

MAP maximum a posteriori

MM Max-Mixture

OSM OpenStreetMap

PID proportional-integral-derivative

QZSS Quasi-Zenith Satellite System

radar radio detection and ranging

RANSAC random sample consensus

RMS root mean square

RRR Realizing, Reversing, Recovering

RTK real-time kinematic

SC Switchable Constraints

SEIF Sparse Extended Information Filter

SLAM simultaneous localization and mapping

SRF sensor reference frame

SVM support vector machine

UTM Universal Transverse Mercator

V-LOAM Vision-lidar Odometry and Mapping

V2X vehicle-to-everything

VO visual odometry

VRF vehicle reference frame

vSLAM visual simultaneous localization and mapping

WGS World Geodetic System

WRF world reference frame

Bibliography

Michael Aeberhard, Sebastian Rauch, Mohammad Bahram, Georg Tanzmeister, Julian Thomas, Yves Pilat, Florian Homm, Werner Huber, and Nico Kaempchen. Experience, Results and Lessons Learned from Automated Driving on Germany's Highways. *IEEE Intelligent Transportation Systems Magazine*, 7 (1), 2015.

Gabriel Agamennoni, Paul Furgale, and Roland Siegwart. Self-tuning M-estimators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

Pratik Agarwal. *Robust Graph-Based Localization and Mapping*. Ph.D. dissertation, University of Freiburg, Germany, 2015.

Pratik Agarwal, Gian D. Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust Map Optimization using Dynamic Covariance Scaling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

Pratik Agarwal, Wolfram Burgard, and Cyrill Stachniss. Survey of Geodetic Mapping Methods: Geodetic Approaches to Mapping and the Relationship to Graph-Based SLAM. *IEEE Robotics & Automation Magazine*, 21(3), 2014a.

Pratik Agarwal, Giorgio Grisetti, Gian D. Tipaldi, Luciano Spinello, Wolfram Burgard, and Cyrill Stachniss. Experimental Analysis of Dynamic Covariance Scaling for Robust Map Optimization Under Bad Initial Estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014b.

Pratik Agarwal, Wolfram Burgard, and Luciano Spinello. Metric Localization using Google Street View. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

Sameer Agarwal, Keir Mierle, and Others. Ceres Solver, 2020. URL `http://ceres-solver.org`.

Naoki Akai, Luis Y. Morales, Takuma Yamaguchi, Eijiro Takeuchi, Yuki Yoshihara, Hiroyuki Okuda, Tatsuya Suzuki, and Yoshiki Ninomiya. Autonomous Driving Based on Accurate Localization Using Multilayer LiDAR and Dead Reckoning. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

Applanix. Datasheet POS LV, 2019. URL `https://www.applanix.com/downloads/products/specs/POS-LV-Datasheet.pdf`. Accessed August 27, 2019.

Michael Baer, Mohamed E. Bouzouraa, Christopher Demiral, Ulrich Hofmann, Stefan Gies, and Klaus Diepold. EgoMaster: A central ego motion estimation for driver assist systems. In *Proceedings of the IEEE International Conference on Control and Automation (ICCA)*, 2009.

Tim Bailey and Hugh Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3), 2006.

Ian Baldwin and Paul Newman. Laser-only road-vehicle localization with dual 2D push-broom LIDARS and 3D priors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Jonathan T. Barron. A General and Adaptive Robust Loss Function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Ioan A. Bârsan, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun. Learning to Localize Using a LiDAR Intensity Map. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2018.

Michael J. Black and Padmanabhan Anandan. The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields. *Computer Vision and Image Understanding*, 63(1), 1996.

Henrik Bohlke. *Verwendung von Fahrbahnmarkierungen zur Verbesserung der graphenbasierten Lokalisierung für das automatisierte Fahren.* Bachelor's thesis, Ostfalia University of Applied Science, 2019.

Claus Brenner. Global Localization of Vehicles Using Local Pole Patterns. In *Lecture Notes in Computer Science*, volume 5748. Springer, Berlin Heidelberg, 2009.

Julia Breßler, Pierre Reisdorf, Marcus Obst, and Gerd Wanielik. GNSS Positioning in Non-line-of-Sight Context - a Survey. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, 2(3), 2017.

Marcus A. Brubaker, Andreas Geiger, and Raquel Urtasun. Map-Based Probabilistic Visual Self-Localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4), 2016.

Mathias Bürki, Lukas Schaupp, Marcin Dymczyk, Renaud Dubé, Cesar Cadena, Roland Siegwart, and Juan Nieto. VIZARD: Reliable Visual Localization for Autonomous Vehicles in Urban Outdoor Environments. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2019.

Cesar Cadena, Luca Carlone, Henry Carillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6), 2016.

Nicholas Carlevaris-Bianco and Ryan M. Eustice. Generic Factor-Based Node Marginalization and Edge Sparsification for Pose-Graph SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

Nicholas Carlevaris-Bianco and Ryan M. Eustice. Conservative Edge Sparsification for Graph SLAM Node Removal. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

Nicholas Carlevaris-Bianco, Michael Kaess, and Ryan M. Eustice. Generic Node Removal for Factor-Graph SLAM. *IEEE Transactions on Robotics*, 30(6), 2014.

Tim Caselitz, Bastian Steder, Michael Ruhnke, and Wolfram Burgard. Monocular Camera Localization in 3D LiDAR Maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Sarah H. Cen and Paul Newman. Precise Ego-Motion Estimation with Millimeter-Wave Radar under Diverse and Challenging Conditions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Sarah H. Cen and Paul Newman. Radar-only ego-motion estimation in difficult settings via graph matching. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Nived Chebrolu, Philipp Lottes, Thomas Läbe, and Cyrill Stachniss. Robot Localization Based on Aerial Images for Precision Agriculture Tasks in Crop

Fields. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Nived Chebrolu, Thomas Läbe, Olga Vysotska, Jens Behley, and Cyrill Stachniss. Adaptive Robust Kernels for Non-Linear Least Squares Problems. *IEEE Robotics and Automation Letters (RA-L)*, 2021.

Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguère, Jens Behley, and Cyrill Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

Han P. Chiu, Stephen Williams, Frank Dellaert, Supun Samarasekera, and Rakesh Kumar. Robust Vision-Aided Navigation Using Sliding-Window Factor Graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

Siddharth Choudhary, Vadim Indelman, Henrik I. Christensen, and Frank Dellaert. Information-based Reduced Landmark SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

Matthew Cornick, Jeffrey Koechling, Byron Stanley, and Beijia Zhang. Localizing Ground Penetrating RADAR: A Step Toward Robust Autonomous Ground Vehicle Localization. *Journal of Field Robotics*, 33(1), 2015.

Mark Cummins and Paul Newman. Probabilistic Appearance Based Navigation and Loop Closing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

Mark Cummins and Paul Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *International Journal of Robotics Research*, 27(6), 2008.

Michael Darms and Hermann Winner. A Modular System Architecture for Sensor Data Processing of ADAS Applications. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2005.

Frank Dellaert. Factor Graphs and GTSAM: A Hands-on Introduction. Technical report, Georgia Institute of Technology, 2012.

Frank Dellaert and Michael Kaess. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *International Journal of Robotics Research*, 25(12), 2006.

Frank Dellaert and Michael Kaess. Factor Graphs for Robot Perception. *Foundations and Trends in Robotics*, 6(1-2), 2017.

John E. Dennis, Jr. and Roy E. Welsch. Techniques for nonlinear least squares and robust regression. *Communications in Statistics - Simulation and Computation*, 7(4), 1978.

Henrik Deusch, Jürgen Wiest, Stephan Reuter, Dominik Nuss, Martin Fritzsche, and Klaus Dietmayer. Multi-Sensor Self-Localization based on Maximally Stable Extremal Regions. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2014.

Ernst D. Dickmanns, Birger Mysliwetz, and Thomas Christians. An Integrated Spatio-Temporal Approach to Automatic Visual Guidance of Autonomous Vehicles. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6), 1990.

Gamini Dissanayake, Paul Newman, Steven Clark, Hugh Durrant-Whyte, and Michael Csorba. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17(3), 2001.

Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part I. *IEEE Robotics & Automation Magazine*, 13(2), 2006.

Kevin Eckenhoff, Liam Paull, and Guoquan Huang. Decoupled, Consistent Node Removal and Edge Sparsification for Graph-based SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Philipp Egger, Paulo V. K. Borges, Gavin Catt, Andreas Pfrunder, Roland Siegwart, and Renaud Dubé. PoseMap: Lifelong, Multi-Environment 3D LiDAR Localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

Nico Engel, Stefan Hoermann, Markus Horn, Vasileios Belagiannis, and Klaus Dietmayer. DeepLocalization: Landmark-based Self-Localization with Deep Neural Networks. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2019.

Ryan M. Eustice, Matthew R. Walter, and John J. Leonard. Sparse Extended Information Filters: Insights into Sparsification. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

Ryan M. Eustice, Hanumant Singh, John J. Leonard, and Matthew R. Walter. Visually mapping the RMS Titanic: Conservative Covariance Estimates for SLAM Information Filters. *International Journal of Robotics Research*, 25 (12), 2006.

Patrick Fleischmann, Thomas Pfister, Moritz Oswald, and Karsten Berns. Using OpenStreetMap for Autonomous Mobile Robot Navigation. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, 2017.

Georgios Floros, Benito van der Zander, and Bastian Leibe. OpenStreetSLAM: Global Vehicle Localization using OpenStreetMaps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

John Folkesson and Henrik Christensen. Graphical SLAM – A Self-correcting Map. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

Wolfgang Förstner. Graphical Models in Geodesy and Photogrammetry. *Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 2013(4), 2013.

Wolfgang Förstner and Bernhard P. Wrobel. *Photogrammetric Computer Vision*, chapter Robust Estimation and Outlier Detection. Springer, 2016.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11(1), 1999.

Friedrich Fraundorfer and Davide Scaramuzza. Visual Odometry Part 2: Matching, Robustness, Optimization, and Applications. *IEEE Robotics & Automation Magazine*, 19(2), 2012.

Udo Frese. A Proof for the Approximate Sparsity of SLAM Information Matrices. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

Udo Frese and Gerd Hirzinger. Simultaneous Localization and Mapping - A Discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2001.

Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. Accepted for publication.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Stuart Geman and Donald E. McClure. Bayesian image analysis: An application to single photon emission tomography. In *Proceedings of the American Statistical Association*, 1985.

Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4), 2010.

Maximilian Harr, Johannes Janosovits, Christoph Stiller, and Sascha Wirges. Fast and Robust Vehicle Pose Estimation by Optimizing Multiple Pose Graphs. In *Proceedings of the International Conference on Information Fusion (FUSION)*, 2018.

Christoph Hertzberg. *A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds*. Master's thesis, University of Bremen, 2008.

Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. Integrating Generic Sensor Fusion Algorithms with Sound State Representations through Encapsulation of Manifolds. *Information Fusion*, 14(1), 2013.

Sebastian Houben, Marcel Neuhausen, Michael Matthias, Robert Kesten, Florian Mickler, and Florian Schuller. Park marking-based vehicle self-localization with a fisheye topview system. *Journal of Real-Time Image Processing*, 16(2), 2015.

Jerry Hsiung, Ming Hsiao, Eric Westman, Rafael Valencia, and Michael Kaess. Information Sparsification in Visual-Inertial Odometry. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

Guoquan Huang, Michael Kaess, and John J. Leonard. Consistent Sparsification for Graph Optimization. In *Proceedings of the European Conference of Mobile Robots (ECMR)*, 2013.

Shoudong Huang and Gamini Dissanayake. A critique of current developments in simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 13(5), 2016.

Peter J. Huber. Robust Estimation of a Location Parameter. *Annals of Mathematical Statistics*, 35(1), 1964.

Peter J. Huber and Elvezio M. Ronchetti. *Robust statistics*. Springer, 2nd edition, 2009.

Constanze Hungar, Jenny Fricke, Stefan Jürgens, and Frank Köster. Detection of Feature Areas for Map-based Localization Using LiDAR Descriptors. In *Proceedings of the Workshop on Positioning, Navigation and Communications (WPNC)*, 2019.

Constanze Hungar, Stefan Jürgens, Daniel Wilbers, and Frank Köster. Map-based Localization with Factor Graphs for Automated Driving using Non-Semantic LiDAR Features. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

International Organization for Standardization. ISO 8855:2011 road vehicles – vehicle dynamics and road-holding ability – vocabulary, 2011. URL `https://www.iso.org/standard/51180.html`.

Prasanth Jeevan, Frank Harchut, Bernhard Mueller-Bessler, and Burkhard Huhnke. Realizing Autonomous Valet Parking with Automotive Grade Sensors. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2018.

Niels Joubert, Tyler G. R. Reid, and Fergus Noble. Developments in Modern GNSS and Its Impact on Autonomous Vehicle Architectures. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2020.

Simon J. Julier. The Stability of Covariance Inflation Methods for SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

Simon J. Julier and Jeffrey K. Uhlmann. A Counter Example to the Theory of Simultaneous Localization and Map Building. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

Stefan Jürgens, Niklas Koch, and Marc-Michael Meinecke. Radar-based Automotive Localization using Landmarks in a Multimodal Sensor Graph-based Approach. In *Proceedings of the International Radar Symposium (IRS)*, 2020.

Lasse Klingbeil, Matthias Nieuwenhuisen, Johannes Schneider, Christian Eling, David Droeschel, Dirk Holz, Thomas Läbe, Wolfgang Förstner, Sven Behnke, and Heiner Kuhlmann. Towards Autonomous Navigation of an UAV-based

Mobile Mapping System. In *Proceedings of the International Conference on Machine Control & Guidance (MCG)*, 2014.

Henrik Kretzschmar and Cyrill Stachniss. Information-theoretic compression of pose graphs for laser-based SLAM. *International Journal of Robotics Research*, 31(11), 2012.

Henrik Kretzschmar, Cyrill Stachniss, and Giorgio Grisetti. Efficient Information-Theoretic Graph Pruning for Graph-Based SLAM with Laser Range Finders. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

Tim Kubertschak, Mirko Mählisch, and Hans-Joachim Wünsche. Towards a unified architecture for mapping static environments. In *Proceedings of the International Conference on Information Fusion (FUSION)*, 2014.

Julius Kümmerle, Marc Sons, Fabian Poggenhans, Tilman Kühner, Martin Lauer, and Christoph Stiller. Accurate and Efficient Self-Localization on Roads using Basic Geometric Primitives. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

Rainer Kümmerle, Bastian Steder, Christian Dornhege, Alexander Kleiner, Giorgio Grisetti, and Wolfram Burgard. Large Scale Graph-based SLAM using Aerial Images as Prior Information. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2009.

Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011a.

Rainer Kümmerle, Bastian Steder, Christian Dornhege, Alexander Kleiner, Giorgio Grisetti, and Wolfram Burgard. Large scale graph-based SLAM using aerial images as prior information. *Autonomous Robots*, 30(1), 2011b.

Sampo Kuutti, Saber Fallah, Konstantinos Katsaros, Mehrdad Dianati, Francis Mccullough, and Alexandros Mouzakitis. A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications. *IEEE Internet of Things*, 5(2), 2018.

Pierre-Yves Lajoie, Siyi Hu, Giovanni Beltrame, and Luca Carlone. Modeling Perceptual Aliasing in SLAM via Discrete-Continuous Graphical Models. *IEEE Robotics and Automation Letters (RA-L)*, 4(2), 2019.

Landesbetrieb Geoinformation und Vermessung. WMS Digitale Orthophotos Hamburg. Licence: "dl-de/by-2-0", available at http://www.govdata.de/dl-de/by-2-0, 2019. URL http://geodienste.hamburg.de/HH_WMS_DOP? Accessed May 17, 2020.

Christian Landsiedel and Dirk Wollherr. Global localization of 3D point clouds in building outline maps of urban outdoor environments. *International Journal of Intelligent Robotics and Applications*, 1(1), 2017.

Dirk Langer. An Integrated MMW Radar System for Outdoor Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1996.

Henning Lategahn, Markus Schreiber, Julius Ziegler, and Christoph Stiller. Urban Localization with Camera and Inertial Measurement Unit. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2013.

Yasir Latif, César Cadena, and José Neira. Robust loop closing over time for posegraph SLAM. *International Journal of Robotics Research*, 32(14), 2013.

Yasir Latif, César Cadena, and José Neira. Robust graph SLAM back-ends: A comparative analysis. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

Kwang W. Lee, Sardha Wijesoma, and Javier Ibañez-Guzmán. A constrained SLAM approach to robust and accurate localisation of autonomous ground vehicles. *Robotics and Autonomous Systems*, 55(7), 2007.

Jesse Levinson and Sebastian Thrun. Robust Vehicle Localization in Urban Environments using Probabilistic Maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

Jesse Levinson and Sebastian Thrun. Automatic Online Calibration of Cameras and Lasers. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2013.

Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-Based Precision Vehicle Localization in Urban Environments. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2007.

Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 32(1), 2016.

Feng Lu and Evangelos Milios. Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4), 1997.

Malin Lundgren, Erik Stenborg, Lennart Svensson, and Lars Hammarstrand. Vehicle Self-localization Using Off-the-Shelf Sensors and a Detailed Map. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2014.

Wei-Chiu Ma, Shenlong Wang, Marcus A. Brubaker, Sanja Fidler, and Raquel Urtasun. Find Your Way by Observing the Sun and Other Semantic Cues. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Wei-Chiu Ma, Ignacio Tartavull, Ioan A. Bârsan, Shenlong Wang, Min Bai, Gellert Mattyus, Namdar Homayounfar, Shrinidhi K. Lakshmikanth, Andrei Pokrovsky, and Raquel Urtasun. Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

Kirk MacTavish and Timothy D. Barfoot. At all Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers. In *Proceedings of the Conference on Computer and Robot Vision (CRV)*, 2015.

Mladen Mazuran, Gian D. Tipaldi, Luciano Spinello, and Wolfram Burgard. Nonlinear Graph Sparsification for SLAM. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2014.

Mladen Mazuran, Wolfram Burgard, and Gian D. Tipaldi. Nonlinear Factor Recovery for Long-Term SLAM. *International Journal of Robotics Research*, 35(1-3), 2016.

Christian Merfels. *Sensor fusion for localization of automated vehicles.* Ph.D. dissertation, University of Bonn, 2018.

Christian Merfels and Cyrill Stachniss. Pose Fusion with Chain Pose Graphs for Automated Driving. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Christian Merfels and Cyrill Stachniss. Sensor Fusion for Self-Localization of Automated Vehicles. *Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85(2), 2017.

Christian Merfels, Tobias Riemenschneider, and Cyrill Stachniss. Pose Fusion with Biased and Dependent Data for Automated Driving. In *Proceedings of the Positioning and Navigation for Intelligent Transportation Systems Conference (POSNAV)*, 2016.

Michael J. Milford and Gordon F. Wyeth. SeqSLAM: Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

Faraz M. Mirzaei, Dimitrios G. Kottas, and Stergios I. Roumeliotis. 3D LIDAR–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *International Journal of Robotics Research*, 31(4), 2012.

Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The Stanford Entry in the Urban Challenge. *Journal of Field Robotics*, 25(9), 2008.

Michael Munz, Mirko Mählisch, Jürgen Dickmann, and Klaus Dietmayer. Probabilistic Modeling of Sensor Properties in Generic Fusion Systems for Modern Driver Assistance Systems. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2010.

Lakshay Narula, Matthew J. Murrian, and Todd. E. Humphreys. Accuracy Limits for Globally-Referenced Digital Mapping Using Standard GNSS. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

Tayyab Naseer and Wolfram Burgard. Deep Regression for Monocular Camera-based 6-DoF Global Localization in Outdoor Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Tayyab Naseer, Wolfram Burgard, and Cyrill Stachniss. Robust Visual Localization Across Seasons. *IEEE Transactions on Robotics*, 34(2), 2018.

Paul Newman, Gabe Sibley, Mike Smith, Mark Cummins, Alastair Harrison, Chris Mei, Ingmar Posner, Robbie Shade, Derik Schroeter, Liz Murphy, Winston Churchill, Dave Cole, and Ian Reid. Navigating, Recognizing and Describing Urban Spaces With Vision and Lasers. *International Journal of Robotics Research*, 28(11-12), 2009.

Wolfgang Niemeier. *Ausgleichungsrechnung – Statistische Auswertemethoden*. de Gruyter, 2nd edition, 2008.

Benjamin Noack, Simon J. Julier, and Uwe D. Hanebeck. Treatment of Biased and Dependent Sensor Data in Graph-based SLAM. In *Proceedings of the International Conference on Information Fusion (FUSION)*, 2015.

Simon Ollander, Friedrich-Wilhelm Bode, and Marcus Baum. Multi-Frequency GNSS Signal Fusion for Minimization of Multipath and Non-Line-of-Sight Errors: A Survey. In *Proceedings of the Workshop on Positioning, Navigation and Communications (WPNC)*, 2018.

Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *International Journal of Robotics Research*, 32(7), 2013.

Oxford Technical Solutions. RT3000, 2020. URL `https://www.oxts.com/products/rt3000/`. Accessed May 19, 2020.

David Pannen, Martin Liebner, Wolfgang Hempel, and Wolfram Burgard. How to Keep HD Maps for Automated Driving Up To Date. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

Jan-Hendrik Pauls, Tobias Strauss, Carsten Hasberg, Martin Lauer, and Christoph Stiller. Can We Trust Our Maps? An Evaluation of Road Changes and a Dataset for Map Validation. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

Fabian Poggenhans, Niels O. Salscheider, and Christoph Stiller. Precise Localization in High-Definition Road Maps for Urban Regions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

François Pomerleau, Francis Colas, and Roland Siegwart. A Review of Point Cloud Registration Algrorithms for Mobile Robotics. *Foundations and Trends in Robotics*, 4(1), 2015.

Ananth Ranganathan, Michael Kaess, and Frank Dellaert. Fast 3D Pose Estimation With Out-of-Sequence Measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

Ananth Ranganathan, David Ilstrup, and Tao Wu. Light-weight Localization for Vehicles using Road Markings. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

Tyler G. R. Reid, Sarah E. Houts, Robert Cammarata, Graham Mills, Siddharth Agarwal, Ankit Vora, and Gaurav Pandey. Localization Requirements for Autonomous Vehicles. *SAE International Journal of Connected and Automated Vehicles*, 2(3), 2019a.

Tyler G. R. Reid, Nahid Pervez, Umair Ibrahim, Sarah E. Houts, Gaurav Pandey, Naveen K. R. Alla, and Andy Hsia. Standalone and RTK GNSS on 30,000 km of North American Highways. In *Proceedings of the International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS)*, 2019b.

Hyunchul Roh, Jinyong Jeong, Younggun Cho, and Ayoung Kim. Accurate Mobile Urban Mapping via Digital Map-Based SLAM. *Sensors*, 16(8), 2016.

David M. Rosen, Julian Mason, and John J. Leonard. Towards Lifelong Feature-Based Mapping in Semi-Static Environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

Lars Rumberg. *Adding landmarks to maps using a graph-based approach.* Master's thesis, University of Hannover, 2018.

Muhamad R. U. Saputra, Andrew Markham, and Niki Trigoni. Visual SLAM and Structure from Motion in Dynamic Environments: A Survey. *ACM Computing Surveys*, 51(2), 2018.

Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry Part 1: The First 30 Years and Fundamentals. *IEEE Robotics & Automation Magazine*, 18 (4), 2011.

Alexander Schaefer, Daniel Büscher, Johan Vertens, Lukas Luft, and Wolfram Burgard. Long-Term Urban Vehicle Localization Using Pole Landmarks Extracted from 3-D Lidar Scans. In *Proceedings of the European Conference of Mobile Robots (ECMR)*, 2019.

Andreas Schindler. *Vehicle Self-Localization with High-Precision Digital Maps.* Ph.D. dissertation, Universität Passau, 2013.

Alexander Schlichting and Claus Brenner. Genauigkeitsuntersuchung zur Lokalisierung von Fahrzeugen mittels Automotive-Laserscannern. In *DGPF Tagungsband 23*, 2014.

Johannes Schneider, Christian Eling, Lasse Klingbeil, Heiner Kuhlmann, Wolfgang Förstner, and Cyrill Stachniss. Fast and Effective Online Pose Estimation and Mapping for UAVs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

Johannes L. Schönberger, Marc Pollefeys, Andreas Geiger, and Torsten Sattler. Semantic Visual Localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Markus Schreiber, Carsten Knöppel, and Uwe Franke. LaneLoc: Lane Marking based Localization using Highly Accurate Maps. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2013.

Frank Schuster, Christoph G. Keller, Matthias Rapp, Martin Haueis, and Christóbal Curio. Landmark based Radar SLAM Using Graph Optimization. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

Mohsen Sefati, Magnus Daum, Björn Sondermann, Kai D. Kreisköther, and Achim Kampker. Improving Vehicle Localization Using Semantic and Pole-Like Landmarks. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2017.

Jacopo Serafin, Edwin Olson, and Giorgio Grisetti. Fast and Robust 3D Feature Extraction from Sparse Point Clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Gabe Sibley. A Sliding Window Filter for SLAM. Technical report, University of Southern California, 2006.

Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding Window Filter with Application to Planetary Landing. *Journal of Field Robotics*, 27(5), 2010.

Sebastian Skibinski. *Extraction, Localization, and Fusion of Collective Vehicle Data*. Ph.D. dissertation, Technische Universität Dortmund, 2019.

Sebastian Skibinski, Frank Weichert, and Heinrich Müller. Parametric Fusion of Complex Landmark Observations Present Within the Road Network by Utilizing Bundle-Adjustment-based Full-SLAM. In *Proceedings of the International Conference on Information Fusion (FUSION)*, 2016.

Isaac Skog and Peter Händel. In-Car Positioning and Navigation Technologies - A Survey. *IEEE Transactions on Intelligent Transport Systems*, 10(1), 2009.

John P. Snyder. *Map projections: A working manual*. U.S. Government Printing Office, 1987.

Marc Sons and Christoph Stiller. Efficient Multi-Drive Map Optimization towards Life-long Localization using Surround View. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018.

Robert Spangenberg, Daniel Goehring, and Raúl Rojas. Pole-based Localization for Autonomous Vehicles in Urban Scenarios. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Cyrill Stachniss and Wolfram Burgard. Particle Filters for Robot Navigation. *Foundations and Trends in Robotics*, 3(4), 2014.

Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. *Springer Handbook of Robotics*, chapter Simultaneous Localization and Mapping. Springer, 2016.

Scott Stephenson. *Automotive Applications of High Precision GNSS*. PhD thesis, University of Nottingham, 2016.

Marek Stess. *Ein Verfahren zur Kartierung und präzisen Lokalisierung mit klassifizierten Umgebungscharakteristiken der Straßeninfrastruktur für selbstfahrende Kraftfahrzeuge*. Ph.D. dissertation, Gottfried Wilhelm Leibniz Universität Hannover, 2017.

Hauke Strasdat, José M. M. Montiel, and Andrew J. Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30(2), 2012.

Jae K. Suhr, Jeungin Jang, Daehong Min, and Ho G. Jung. Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments. *IEEE Transactions on Intelligent Transport Systems*, 18(5), 2017.

Niko Sünderhauf. *Robust optimization for Simultaneous Localization and Mapping*. Ph.D. dissertation, Chemnitz University of Technology, 2012.

Niko Sünderhauf and Peter Protzel. Switchable Constraints for Robust Pose Graph SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Niko Sünderhauf and Peter Protzel. Switchable Constraints vs. Max-Mixture Models vs. RRR - A Comparison of Three Approaches to Robust Pose Graph SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

Niko Sünderhauf, Sareh Shirazi, Adam Jacobson, Feras Dayoub, Edward Pepperell, Ben Upcroft, and Michael J. Milford. Place Recognition with ConvNet Landmarks: Viewpoint-Robust, Condition-Robust, Training-Free. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2015.

Duy-Nguyen Ta, Nandan Banerjee, Stephen Eick, Scott Lenser, and Mario E. Munich. Fast Nonlinear Approximation of Pose Graph Node Marginalization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(16), 2017.

Tim Y. Tang, Daniele De Martini, Dan Barnes, and Paul Newman. RSL-Net: Localising in Satellite Images From a Radar on the Ground. *IEEE Robotics and Automation Letters (RA-L)*, 5(2), 2020a.

Tim Y. Tang, Daniele De Martini, Shangzhe Wu, and Paul Newman. Self-Supervised Localisation between Range Sensors and Overhead Imagery. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, 2020b.

Tesla, Inc. Introducing Navigate on Autopilot, 2018. URL `https://www.tesla.com/blog/introducing-navigate-autopilot`. Accessed February 19, 2021.

Peter J. G. Teunissen and Oliver Montenbruck, editors. *Springer Handbook of Global Navigation Satellite Systems*. Springer, 2017.

Sebastian Thrun and Michael Montemerlo. The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *International Journal of Robotics Research*, 25(5-6), 2006.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The Robot that Won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9), 2006.

Sadayuki Tsugawa, Teruo Yatabe, Takeshi Hirose, and Shuntetsu Matsumoto. An automobile with artificial intelligence. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, 1979.

u-blox. NEO-M8L series, 2020. URL `https://www.u-blox.com/en/product/neo-m8l-series`. Accessed November 08, 2020.

U.S. Government - National Coordination Office for Space-Based Positioning, Navigation, and Timing. GPS Accuracy, 2018. URL `https://www.gps.gov/systems/gps/performance/accuracy/`. Accessed June 26, 2018.

Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep Auxiliary Learning for Visual Localization and Odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Joan Vallvé, Joan Solà, and Juan Andrade-Cetto. Factor descent optimization for sparsification in graph SLAM. In *Proceedings of the European Conference of Mobile Robots (ECMR)*, 2017.

Joan Vallvé, Joan Solà, and Juan Andrade-Cetto. Graph SLAM sparsification with populated topologies using factor descent optimization. *IEEE Robotics and Automation Letters (RA-L)*, 3(2), 2018.

Joan Vallvé, Joan Solà, and Juan Andrade-Cetto. Pose-graph SLAM sparsification using factor descent. *Robotics and Autonomous Systems*, 119(1), 2019.

Sudha Vana, John Aggrey, Sunil Bisnath, Rodrigo Leandro, Landon Urquhart, and Paola Gonzalez. Analysis of GNSS correction data standards for the automotive market. *NAVIGATION*, 66(3), 2019.

Veloyne Lidar. Velodyne VLP-32C, 2020. URL `https://velodynelidar.com/products/ultra-puck/`. Accessed November 08, 2020.

John Vial, Hugh Durrant-Whyte, and Tim Bailey. Conservative Sparsification for Efficient and Consistent Approximate Estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

Antoni R. Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios. *IEEE Robotics and Automation Letters (RA-L)*, 3(2), 2018.

Volkswagen AG. The assistance systems of the new Arteon - Interactive technologies look ahead for safety, 2017. URL `https://www.volkswagen-newsroom.com/en/stories/the-assistance-systems-of-the-new-arteon-interactive-technologies-look-ahead-for-safety-2268`. Accessed February 19, 2021.

Volkswagen AG. DB2019AL00706, 2019. URL `https://www.volkswagen-newsroom.com/de/pressemitteilungen/volkswagen-faehrt-vollautomatisiert-in-hamburg-4797`. Accessed November 08, 2020.

Andrej Štern and Anton Kos. Positioning Performance Assessment of Geodetic, Automotive, and Smartphone GNSS Receivers in Standardized Road Scenarios. *IEEE Access*, 6(1), 2018.

Arun Vydhyanathan and Giovanni Bellusci. The Next Generation Xsens Motion Trackers for Industrial Applications. Technical report, Xsens, 2018.

Olga Vysotska. *Visual Place Recognition in Changing Environments*. Ph.D. dissertation, University of Bonn, 2019.

Olga Vysotska and Cyrill Stachniss. Exploiting Building Information from Publicly Available Maps in Graph-Based SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016a.

Olga Vysotska and Cyrill Stachniss. Lazy Data Association For Image Sequences Matching Under Substantial Appearance Changes. *IEEE Robotics and Automation Letters (RA-L)*, 1(1), 2016b.

Olga Vysotska and Cyrill Stachniss. Improving SLAM by Exploiting Building Information from Publicly Available Maps and Localization Priors. *Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85(1), 2017.

Olga Vysotska and Cyrill Stachniss. Effective Visual Place Recognition Using Multi-Sequence Maps. *IEEE Robotics and Automation Letters (RA-L)*, 4(2), 2019.

Matthew R. Walter, Ryan M. Eustice, and John J. Leonard. A Provably Consistent Method for Imposing Sparsity in Feature-Based SLAM Information Filters. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2007.

Liang Wang, Yihuan Zhang, and Jun Wang. Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR. In *Proceedings of the IFAC World Congress*, 2017.

Xinkai Wei, Ioan A. Bârsan, Shenlong Wang, Julieta Martinez, and Raquel Urtasun. Learning to Localize Through Compressed Binary Maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Daniel Wilbers, Stefan Jürgens, David Perdomo Lopez, Christian Merfels, Constanze Hungar, Bernd Rech, Thilo Schaper, and Niklas Koch. Verfahren zur Aktualisierung einer Umgebungskarte, Vorrichtung für die fahrzeugseitige Durchführung von Verfahrensschritten des Verfahrens, Fahrzeug, Vorrichtung für die zentralrechnerseitige Durchführung von Verfahrensschritten des Verfahrens sowie computerlesbares Speichermedium. Patent application at Deutsches Patent- und Markenamt, Germany, DE 10 2018 118 215.5, 2018a.

Daniel Wilbers, Stefan Jürgens, David Perdomo Lopez, Christian Merfels, Constanze Hungar, Bernd Rech, Thilo Schaper, and Niklas Koch. Positionsbestimmungssystems für eine mobile Einheit. Patent application at Deutsches Patent- und Markenamt, Germany, DE 10 2018 117 660.0, 2018b.

Daniel Wilbers, Stefan Jürgens, David Perdomo Lopez, Christian Merfels, Constanze Hungar, Bernd Rech, Thilo Schaper, and Niklas Koch. Verfahren zur Schätzung der Lokalisierungsgüte bei der Eigenlokalisierung eines Fahrzeuges, Vorrichtung für die Durchführung von Verfahrensschritten des Verfahrens, Fahrzeug sowie Computerprogramm. Patent application at Deutsches Patent- und Markenamt, Germany, DE 10 2018 118 220.1, 2018c.

Daniel Wilbers, Stefan Jürgens, David Perdomo Lopez, Christian Merfels, Constanze Hungar, Bernd Rech, Thilo Schaper, and Niklas Koch. Positionsbestimmungssystem und Verfahren zum Betreiben eines Positionsbestimmungssystems für eine mobile Einheit. Patent application at Deutsches Patent- und Markenamt, Germany, DE 10 2018 133 461.3, 2018d.

Daniel Wilbers, Christian Merfels, and Cyrill Stachniss. Localization with Sliding Window Factor Graphs on Third-Party Maps for Automated Driving. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019a.

Daniel Wilbers, Christian Merfels, and Cyrill Stachniss. A Comparision of Particle Filter and Graph-based Optimization for Localization with Landmarks in Automated Vehicles. In *Proceedings of the IEEE International Conference on Robotic Computing (IRC)*, 2019b.

Daniel Wilbers, Lars Rumberg, and Cyrill Stachniss. Approximating Marginalization with Sparse Global Priors for Sliding Window SLAM-Graphs. In *Proceedings of the IEEE International Conference on Robotic Computing (IRC)*, 2019c.

Daniel Wilbers, Henrik Bohlke, Stefan Jürgens, Christian Merfels, Constanze Hungar, Bernd Rech, and Niklas Koch. Verfahren zum Bewerten einer digitalen Karte, sowie Bewertungssystem. Patent application at Deutsches Patent- und Markenamt, Germany, DE 10 2020 115 743.6, 2020a.

Daniel Wilbers, Henrik Bohlke, Stefan Jürgens, Christian Merfels, Constanze Hungar, Bernd Rech, and Niklas Koch. Verfahren zum Beurteilen einer Genauigkeit einer Positionsbestimmung einer Landmarke, sowie Bewertungssystem. Patent application at Deutsches Patent- und Markenamt, Germany, DE 10 2020 115 746.0, 2020b.

Dan Withers and Paul Newman. Modeling Scene Change for Large-Scale Long Term Laser Localisation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Ryan W. Wolcott and Ryan M. Eustice. Visual Localization within LIDAR Maps for Automated Urban Driving. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

Ryan W. Wolcott and Ryan M. Eustice. Fast LIDAR localization using Multiresolution Gaussian Mixture Maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

Cong Wu, Tiffany A. Huang, Maximilian Muffert, Thilo Schwarz, and Johannes Graeter. Precise Pose Graph Localization with Sparse Point and Lane Features. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Heng Yang, Pasquale Antonante, Vasileios Tzoumas, and Luca Carlone. Graduated Non-Convexity for Robust Spatial Perception: From Non-Minimal Solvers to Global Outlier Rejection. *IEEE Robotics and Automation Letters (RA-L)*, 5(2), 2020a.

Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and Certifiable Point CloudRegistration. *IEEE Transactions on Robotics*, 2020b. Accepted for publication.

Sheng Yang, Xiaoling Zhu, Xing Nian, Lu Feng, Xiaozhi Qu, and Teng Ma. A robust pose graph approach for city scale LiDAR mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A Survey of Autonomouos Driving: Common Practices and Emerging Technologies. *IEEE Access*, 8(1), 2020.

Ji Zhang and Sanjiv Singh. Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

Zhengyou Zhang. Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing*, 15(1), 1997.

Julius Ziegler, Thao Dang, Uwe Franke, Henning Lategahn, Philipp Bender, Markus Schreiber, Tobias Strauss, Nils Appenrodt, Christoph G. Keller, Eberhard Kaus, Christoph Stiller, Ralf G. Herrtwich, and et al. Making Bertha Drive – An Autonomous Journey on a Historic Route. *IEEE Intelligent Transportation Systems Magazine*, 6(2), 2014.

Florian Zimmermann, Christian Eling, and Heiner Kuhlmann. Investigations on the Influence of Antenna Near-field Effects and Satellite Obstruction on the Uncertainty of GNSS-based Distance Measurements. *Journal of Applied Geodesy*, 10(1), 2015.

List of Figures

List of Tables

List of Algorithms