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

Sensor fusion for localization of automated vehicles

von

Christian Merfels

aus

Leverkusen, Germany

Bonn 2018

Referent:
Prof. Dr. Cyrill Stachniss, Friedrich-Wilhelms-Universität Bonn

1. Korreferent:
Prof. Dr.-Ing. Heiner Kuhlmann, Friedrich-Wilhelms-Universität Bonn

Tag der mündlichen Prüfung: 6. Dezember 2018

Angefertigt mit Genehmigung der Landwirtschaftlichen Fakultät der Universität Bonn

Abstract

Automated vehicles need to precisely know where they are at all times to be able to make informed driving decisions. Therefore, multiple localization systems are typically installed on such vehicles to provide redundant position estimates based on different sensors. Thus, an important task is the fusion of these position estimates into a single estimate. The goal of this thesis to develop a new approach to solve this sensor fusion problem in a generic way to achieve high modularity, interchangeability, and extensibility, while at the same time assuring high precision, robustness, and availability.

Generic approaches to sensor fusion for localization systems face the difficulty that only general assumptions can be made about their input data. These generic assumptions make it complicated to model the error of each input source differently.

We approach this challenge by presenting a novel layer architecture that can be modularly adapted. The core of our generic fusion approach is an optimization method that combines all available position and odometry measurements. We formulate a sliding window pose graph over these measurements to estimate the most probable trajectory of the vehicle. In a preprocessing sublayer, the measurements are adjusted so that different common error characteristics are either reduced or can be taken into account in the estimation process. These include systematic, autocorrelated, and cross-correlated errors as well as outliers. We derive different preprocessing modules for each of these error modes.

In this thesis, we extend the pose graph model to represent the effects of autocorrelated errors and marginalization. We implement our approach and evaluate it using simulated data as well as data gathered on real prototype vehicles. In experiments, we show that the estimation method scales from a filtering-based to a batch solution depending on the available computational resources. In addition, we demonstrate that our preprocessing modules reduce the effects of the described error characteristics. Overall, we develop a generic fusion of position estimates, which is a key component of automated vehicles.

Zusammenfassung

Automatisierte Fahrzeuge müssen jederzeit genau wissen, wo sie sich befinden, um fundierte Fahrentscheidungen treffen zu können. Deshalb sind auf solchen Fahrzeugen üblicherweise mehrere Lokalisierungssysteme installiert, um redundante Positionsschätzungen auf Basis unterschiedlicher Sensoren zu ermöglichen. Hieraus ergibt sich die zentrale Aufgabe der Fusion von diesen Positionsinformationen in eine einzige Schätzung. Das Ziel dieser Arbeit ist es, diese Fusion auf eine generische Art und Weise zu gestalten, um eine hohe Modularität, Austauschbarkeit und Erweiterbarkeit zu erzielen und gleichzeitig eine hohe Genauigkeit, Robustheit und Verfügbarkeit zu gewährleisten.

Solche generischen Fusionsansätze für Lokalisierungssysteme bergen die Schwierigkeit, dass nur allgemeine Annahmen über die zu fusionierenden Eingangsdaten getroffen werden können. Diese generischen Annahmen erschweren eine auf jede Eingangsquelle abgestimmte Fehlermodellierung.

Um dennoch eine differenzierte Verarbeitung zu ermöglichen, stellen wir eine neue Schichtenarchitektur vor, die modular angepasst werden kann. Der Kern unseres generischen Fusionsansatzes bildet ein Optimierungsverfahren, in dem alle vorhandenen Positions- und Eigenbewegungsmessungen kombiniert werden. Dazu konstruieren wir einen Posengraphen über alle Messungen im letzten Zeitfenster, um so die wahrscheinlichste gefahrene Trajektorie zu schätzen. In einer Vorverarbeitungsschicht werden die Messdaten so angepasst, dass verschiedene in der Praxis häufig auftretende Fehlercharakteristiken entweder reduziert oder im Schätzverfahren besser berücksichtigt werden. Diese beinhalten systematische, autokorrelierte und kreuzkorrelierte Fehler sowie Ausreißer. Für diese Fehlerbilder leiten wir Verfahren zu ihrer Vorverarbeitung her.

Im Rahmen dieser Arbeit entwickeln wir das Modell des Posengraphen weiter, um darin die Effekte der Marginalisierung und autokorrelierter Fehler darstellen zu können. Wir implementieren unseren Ansatz und evaluieren ihn mit Hilfe simulierter und auf realen Prototypen aufgezeichneter Daten. In Experimenten zeigen wir, dass das Schätzverfahren von einer filter-basierten bis zur Batch-Lösung skaliert in Abhängigkeit der zur Verfügung stehenden Rechenkapazitäten. Außerdem weisen wir nach, dass die Vorverarbeitungsmodule die Auswirkungen der beschriebenen Fehlercharakteristiken effektiv reduzieren. Insgesamt entwickeln wir eine generische Fusion von Positionsschätzungen, die eine zentrale Komponente von automatisierten Fahrzeugen ist.

Acknowledgments

I wrote this thesis as external Ph.D. student of the University of Bonn at Volkswagen Group Research. I am very grateful to my advisor Prof. Dr. Cyrill Stachniss for giving me the opportunity to work with him in the Photogrammetry & Robotics Lab. It is not easy to integrate an external Ph.D. student into a research group but he did a great job at it. Our discussions, his ideas and his support vastly improved my work. To put it in a nutshell, I truly appreciate his confidence in me and that he always treated me like a colleague.

I would also like to thank all the other people at the lab, including Igor Bogoslavskyi, Olga Vysotska, Lorenzo Nardi, Emanuele Palazzolo, Susanne Wenzel, and Birgit Klein. They made me feel very welcome and gave me an insight into their life and work at the university. I especially enjoyed our time at the IROS conference in Korea.

My special thanks go to my friends Moritz Schack, Bernd Rech, and Thorsten Hehn for sharing their experiences and guidances on how to finish a Ph.D. thesis. My thanks to Thilo Schaper, Niklas Koch, Daniel Wilbers, Marek Stess, Constanze Hungar, Stefan Jürgens, and Andreas Kwoczek for being supportive and helpful friends. Special thanks to Stefan Jürgens and Bernd Rech for proof-reading this manuscript.

Equally, I would like to to thank my colleagues and supervisors at Volkswagen Group Research in the department for Automated Driving. They provided a nice and friendly working atmosphere and greatly supported my work.

My thanks to Tile Lefherz, Marcel Geppert, Mareike Ploog, and Lukas Fröhlich for their collaboration and for bringing a fresh breeze into our group.

Lastly but perhaps most importantly, I would like to thank my family and friends for always being there for me. I would not have finished this work without your support.

Contents

Disclaimer

1. Introduction

Transportation and mobility are central elements of our everday lifes and important aspects of the modern society. Automated vehicles have the potential to reduce the number of accidents, to relieve the driver from the chore of driving, to decrease the pollution, and to increase driving comfort. Thus, there is an active field of research concerned with developing automated vehicles. This thesis focuses on one part of it, namely the estimation of the automated vehicle's location and orientation from sensor data.

1.1. Estimating the location and orientation of the vehicle

Automated vehicles are a particular kind of mobile robot: they sense their environment, interpret this information, plan navigation actions, and finally apply these actions to drive through their environment. To this end, they act in specific environments like outdoors on roads or indoors in parking garages. They also face specific challenges such as other road users. As for any mobile robot, the ability of an automated vehicle to determine its own position and heading (often called *pose*) is a key component for the fulfillment of its tasks. This self-localization capability opens up the possibility to use geospatial information stored in maps. Maps provide a priori information that, depending on the map content, can ease certain navigation decisions but also perception, path planning, and collision avoidance.

Due to the importance of localization, many different techniques for pose estimation have been developed. Some of them provide information about the global pose of the vehicle (e.g., based on Global Navigation Satellite System (GNSS)), and some provide information about the pose of the vehicle relative to the last time step (so-called *odom-*

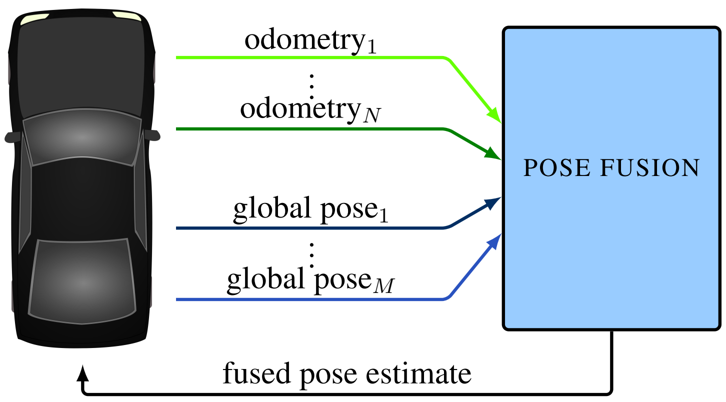


Figure 1.1.: Overview of the general concept of a pose fusion. Multiple localization systems or positioning sensors serve as input to compute the fused pose estimate.

etry, e.g., inertial measurement unit (IMU)). Typically, multiple of these *localization systems* (or *pose sources*) are working in parallel on an automated vehicle to ensure redundancy. In this thesis, we focus on merging this information to provide an accurate, single estimate of the current vehicle's pose.

Merging information from multiple sources falls into the area of multi-sensor data fusion. This field is sometimes also referred to as sensor fusion and has become a synonym for state estimation. The term "sensor fusion" has many definitions in the literature. In this thesis, we stick to the definition of Elmenreich (2002) who defines it as *"[...] the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually"*. In the context of automated driving, all information is about the pose of the vehicle, and we therefore render the terminology more precisely by referring to sensor fusion for pose estimation as *pose fusion*. Figure 1.1 illustrates this concept by showing that multiple localization systems (odometry and global pose sources) serve as input for a pose fusion which computes a unified pose estimate as output. Multi-sensor data fusion techniques combine information from several sources to provide information that is in a sense more valuable than that of a single source. In what way does this gain in value manifest itself? For an automated vehicle, there are several aspects how we would like to benefit from a pose fusion.

Pose fusion has the potential to increase the availability of the overall localization solution. Localization systems are usually tailored to a specific sensor set, which includes GNSS, vision, or light detection and ranging (lidar) sensors. Each of these sensors has individual failure modes such as satellite-denied regions for GNSS or darkness for visual systems. In these situations, the localization system's performance degrades substantially or fails entirely to provide a reasonable estimate of the vehicle's pose. Another class of global pose sources that are not always available are map-based localization systems. They aim at aligning the sensor information to a given map and in this way estimate the pose of the vehicle in the map. Naturally, they are limited to areas in which maps are available.

A benefit of combining multiple localization systems is that fallback localization modules can be integrated. They are only relevant whenever the main localization modules fail. Suppose we have a high-accuracy map-based localization based on camera images. This visual localization system can fail in case of darkness or severe lighting conditions. In practice, it is often possible to additionally obtain a pose estimate with a GNSS receiver. In case of a failure of the visual high-accuracy localization system, we would still like to be able to provide a pose estimate, albeit with a higher uncertainty. In this setup, the pose estimates of the GNSS receiver would always be included in the pose fusion process, but only become dominant whenever the visual system fails. Pose fusion allows the combination of orthogonal sensing modalities and therefore increases the reliability and versatility of the localization output. It also allows us to cope with (temporary) failure of pose sources.

Furthermore, the seamless transition between scenarios for which different localization systems are valid becomes possible. A localization system based on detecting fiducial markers with a camera can provide a pose estimate in a specifically prepared parking garage, for example. Outside of the parking garage, a conventional GNSS receiver might serve as the primary localization system, see Figure 1.2. Neither localization module is able to estimate a pose in both scenarios, but the multi-sensor fusion is able to regularly provide a pose output.

Our fusion approach effectively glues partial trajectory estimates together and explicitly considers all pose sources, making it unnecessary to switch abruptly between them in different scenarios. Such a switching might be particularly difficult when the exact system boundaries are unknown, e.g., where does the GNSS receiver start producing pose estimates, and what to do in situations where both or none of the systems provide

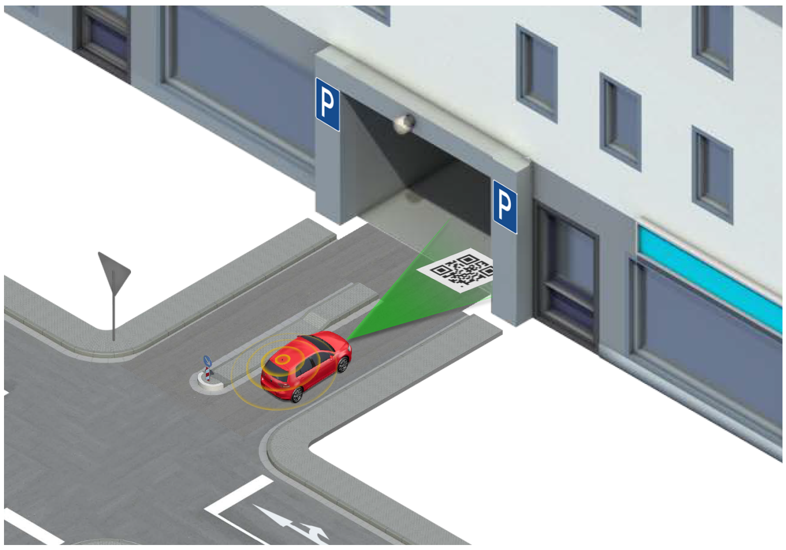


Figure 1.2.: An automated vehicle enters a parking garage. It has GNSS reception outside of the parking garage and localizes itself within the garage with the help of fiducial markers that are detected by a camera. A pose fusion opens up the possibility of a seamless transition between both scenarios.

a pose estimate. The use of a pose fusion hides these difficulties from the application.

In an automated vehicle, there is potentially a multitude of components that rely on the current pose as an input, e.g., the path planner or the navigation module. Feeding data streams from multiple localization systems directly to these application modules means that each one of them has to make sense of this set of streams individually. By adding a fusion layer this task is centralized and its output is standardized independently of the underlying localization systems. This layer hides the specific characteristics of localization modules by providing a single interface. In this architecture, it is easier to change the underlying combination of sensors and localization systems because this does not require changes in the application layer. This happens rather frequently in an environment where the sensors and localization systems themselves are also subject of research.[1]

Automated vehicles and advanced driver assistance system (ADAS) rely on the precise knowledge of the vehicle's pose. The fusion of several sources of information has

[1]During the development of this thesis, only a single pose source remained unchanged (the wheel odometry). All other localization systems where either exchanged, substantially altered, or removed; and some were newly added.

the potential to increase the accuracy of the estimated pose. Depending on the accuracy gain, previously unfeasible applications might become possible or their performance could be improved.

Closely related to the improvement of accuracy is the reduction of uncertainty. Speaking from a state estimation perspective, it is well-known that using multiple measurements to estimate a single state reduces the uncertainty about this state. For a pose fusion, the fused estimate has thus a lower uncertainty than the input poses.

In total, using a pose fusion aims to

- increase the availability of the vehicle's pose estimate,
- increase its reliability against failure compared to using a single localization system,
- enable the transition between different scenarios in which some sensors do not work,
- reduce the system complexity,
- reduce the uncertainty of the pose estimate,
- and increase its accuracy.

1.2. Generic pose fusion

In Section 1.1 we highlight the potential benefits of an approach to pose fusion. Today's pose fusions for vehicles are tailored to specific sensor sets and localization systems (e.g., combinations of GNSS and IMU). In this work, in contrast, we are investigating how to combine information of pose sources, which we treat in a *generic* way. We do not derive a pose fusion that is tailored to a *specific* set of sensors or pose sources. Instead, we approach the problem of how to construct a pose fusion that can cope with a set of pose sources for which we only make generic assumptions, thus leading to the problem of *generic pose fusion*.

If we can approach the pose fusion problem on this generic level, while still maintaining the advantages outlined before, then we are able to use the same pose fusion for different sets of pose sources. In practice, this happens often as different vehicles have different sensor setups such that also the localization systems for these vehicles differ.

The proposed loose coupling of localization and fusion promotes a modular architecture. It eliminates the need to design new fusion schemes or extend existing ones if a new pose source is to be added. Also, we can easily incorporate information from pose sources for which source code or deep knowledge of their internal concepts are unavailable.

This leads us to the key question of this work: how can we design a pose fusion that treats all pose sources in a generic way?

1.3. Aim and scope

Our aim is strictly aligned to the key problem of this thesis: the goal is to derive a generic pose fusion system and apply it to an automated vehicle. To define what is part of this investigation, and what is not, we present the limitations and system boundaries of our pose fusion. Additionally, we derive requirements for a generic pose fusion concept.

A generic fusion concept shall be independent of the specific type of input localization systems. This immediately sets a first scope for this work: we will treat input data in a generic manner and not build a fusion specific to a certain set of input localization systems.

Note that we do not assume that all pose sources behave exactly the same way. Instead, we argue that we can group them into classes such that we can apply class-specific preprocessing techniques to increase their conformity with our generic assumptions. Note that, in the spirit of generic pose fusion, it is our goal to form these classes as broad and as less specific to a single source as possible.

Our pose fusion tries to best combine pose data and not to compute a pose estimate from scratch. This decoupling of the localization and fusion means that it can only produce an output whenever (at least some) input data for the fusion is available. Formulating this as a requirement, we require an availability of 100%: the pose fusion has to always provide the best possible pose, under all conditions, as long as any input data is available.

Moreover, to be of use for the other ADAS modules, the output of the pose fusion has to be recent. In this thesis, the *latency* of a pose estimate is the temporal difference between when it becomes available and for what time it is valid. In practice, the latency of the output of the pose fusion depends among others on the available pose sources. In the evaluations of this thesis, there is almost always odometry data available with a low latency. Therefore, we demand a latency of our pose fusion in the same order of magnitude.

The pose fusion has to run online on limited hardware in automated vehicles. It has to work on several, different vehicles with acceptable configuration effort. Therefore, we require a scalable approach (in terms of runtime performance) that gets better with more computational resources, but it also has to work on hardware with less performance. The pose fusion would ideally adapt its own configuration to the available hardware to minimize the configuration effort.

As we perform generic fusion, it is a key challenge that we cannot rely on any timing requirements for the input data, i.e., assume that the data arrives instantly, is strictly ordered, without packet loss, or arrives with constant frequency. The only exception to this is that we require the data to be correctly timestamped for all pose sources.

The pose fusion works with three degrees of freedom which are the location and orientation of the vehicle. We do not estimate all six degrees of freedom for two reasons. First, this is a common assumption for automated vehicles and they generally do not rely on all six degrees of freedom. Second, all available input sources only provide pose estimates with three degrees of freedom. However, we require that the concept of the pose fusion is in principle extensible to six degrees of freedom for possible future enhancements. This would enable its application on systems like drones or underwater robots.

The accuracy of the pose fusion is of great importance. However, we cannot impose a strict accuracy requirement as it strongly depends on the quality of the input data. Instead, we require that the pose fusion accurately converges towards the statistically optimal result. We require the pose fusion to make optimal use of the available information to provide the most accurate output possible.

Finnaly, the pose fusion has to be able to provide estimates about the uncertainty of its output. This is important for several applications that rely on the pose fusion data and use probabilistic techniques.

Now that we have defined the scope of this work, we can formulate the goal that we pursue more clearly. The aim of this work is to develop a generic pose fusion that satisfies the following requirements:

- the output of the pose fusion has to be always available whenever input data is available,
- the latency of the pose fusion has to be in the order of the most recent input data,
- the pose fusion has to be an online approach,
- the pose fusion has to scale its computational requirements with the available hardware,
- in principle, the pose fusion concept has to be extensible to estimation in full six degrees of freedom,
- common error characteristics of the input pose estimates have to be modeled, reduced, or eliminated,
- the estimation of the pose fusion has to converge to the statistically optimal result,
- the pose fusion needs to provide reasonable estimates of its uncertainty.

1.4. Contributions

The main contribution of this thesis is an online pose fusion algorithm that makes little assumptions about the types of the underlying localization systems. Our approach is structured into a layered architecture such that we can reuse modules if appropriate. We divide it into a core estimator and a set of preprocessing modules, which transform the input data for optimal use by the core estimator. Due to its generic nature, the pose fusion is applicable to automated vehicles, mobile robots, and other systems having access to multiple pose sources.

The core estimator avoids overconfidence by performing delayed marginalization. It is formulated as a sliding window graph-based optimization that leads to a maximum likelihood (ML) estimate over the joint probability of vehicle poses in the current window. It converges to the online ML estimate for increasing sizes of the sliding window. Different parametrizations make it possible to scale from the Iterated Extended Kalman filter (IEKF) to the batch solution and thus to balance runtime versus accuracy. This enables the use with different hardware configurations. We create an efficient estimator that has linear runtime and a constant memory complexity in the size of its state. It retains the sparseness of its optimization problem over time, making the approach independent of the duration of operation. This is achieved by focusing on creating so-called chain pose graphs. In addition, we present a technique which allows it to adapt its computational load to the available resources at runtime by parametrizing its algorithm accordingly.

We develop methods to gain an understanding of the effects of state marginalization and autocorrelation in the graph-based formulation. This leads to new nodes and edges which are the prior node and factors to represent autocorrelated error terms of both local and global pose measurements. These are applicable to other methods that use a similar formulation, i.e., many graph-based Simultaneous Localization and Mapping (SLAM) algorithms.

The preprocessing sublayer developed in this work consists of four modules. They serve to model, reduce, or eliminate common error characteristics. The first module estimates time-varying biases of global poses. It compares input poses from different sources to determine biases. The second module reduces the influence of outliers. It determines outliers by comparing them to a map and increases their uncertainty if necessary. The third module treats input poses from sources with cross-correlated noise relying on Covariance Intersection (CI). The fourth module models autocorrelated errors by inflating the covariance matrices of the corresponding pose estimates. These four modules are valuable for approaches other than generic pose fusion and easily implementable without the core estimator.

Parts of this thesis have been published in the following conference proceedings and journal articles:

- 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)*, pages 3116–3123, 2016,

- 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 (POS-*

NAV ITS), 2016. ISSN: 2191-8287,

- Christian Merfels and Cyrill Stachniss. Sensor fusion for self-localization of automated vehicles. *Journal of Photogrammetry, Remote Sensing, and Geoinformation Science (PFG)*, 85(2):113–126, 2017.

The work in Section 5.6 on recovering the covariance matrix of the fused output as presented was done in tight collaboration with Mareike Ploog and was originally addressed in her Master's thesis (Ploog, 2017) that I co-supervised. Additionally, the preprocessing technique for scaling a covariance matrix based on map information as presented in Section 6.2 was done in tight collaboration with Lukas Fröhlich and is addressed in his Master's thesis (Fröhlich, 2017) that I co-supervised. The same holds for the preprocessing technique for decreasing the influence of autocorrelated errors as presented in Section 6.4.

The following patents have been filed during the development and in the context of this thesis:

- Christian Merfels and Moritz Schack. Fusion von Positionsdaten mittels Posen-Graph. Patent application at Deutsches Patent- und Markenamt, Germany, October 2015. DE 10 2015 219 577.5. This refers to the general concept of the core estimator presented in Section 5.

- Christian Merfels and Moritz Schack. Marginalisieren eines Posen-Graphen. Patent application at Deutsches Patent- und Markenamt, Germany, March 2016. DE 10 2016 205 193.8. This refers to the prior node presented in Section 5.3.

- Christian Merfels. Kompensation von Fehlern in Absolut-Positionsdaten bei der Schätzung der Eigenposition. Patent application at Deutsches Patent- und Markenamt, Germany, October 2016. DE 10 2016 220 593.5. This refers to the bias estimation module presented in Section 6.1.

- Christian Merfels, Lukas Fröhlich, Bernd Rech, Thilo Schaper, Niklas Koch, and Daniel Wilbers. Verfahren, Vorrichtung und computerlesbares Speichermedium mit Instruktionen zur Schätzung einer Pose eines Kraftfahrzeugs. Patent application at Deutsches Patent- und Markenamt, Germany, April 2017b. DE 10 2017 108 107.0. This refers to the map-based outlier handling presented in Section 6.2.

- Christian Merfels, Lukas Fröhlich, and Bernd Rech. Verfahren zur Datenfusion eines Datensatzes, entsprechende Recheneinheit und Fahrzeug welches mit einer entsprechenden Recheneinheit ausgestattet ist sowie Computerprogramm. Patent application at Deutsches Patent- und Markenamt, Germany, April 2017a. 10 2017 108 130.5. This refers to the modeling of autocorrelated errors presented in Section 6.4.

1.5. Overview of the thesis

This thesis is organized as follows. First, we review the state of the art in Chapter 2 for the different methods addressed in this thesis. Subsequently, we present fundamental techniques and background knowledge in Chapter 3. The main architecture for our generic pose fusion is presented in Chapter 4. Its key component is the design of a layered fusion architecture in which we split up the fusion layer into a core estimator and a sublayer of preprocessing techniques. In Chapter 5, we design and develop the core estimator concept.

In Chapter 6, we address valuable techniques for preprocessing the input data for the core estimator. We describe in Section 6.1 a generic preprocessing technique that focuses on decreasing the influence of biases on pose estimates. In Section 6.2, we present a method for outlier handling. In Section 6.3, we address the problem of fusing data between pose sources whose error is correlated. Section 6.4 describes how to treat autocorrelated noise of a pose source in such a way that the estimator can ignore this characteristic. In Chapter 7, we provide separate evaluations for the core estimator, the individual techniques of the preprocessing sublayer, and for the entire pose fusion that combine all contributions.

Without getting into the details of the layer architecture of our pose fusion just yet, we already present parts of it here for the sake of using it as a guidance on where to find the different parts of this thesis. Figure 1.3 shows the heart of the architecture and the mapping between the topic and the respective chapter number.

Generally, if a reader is only interested in a particularly topic of this thesis, then we invite the reader to start with the corresponding section of related work in Chapter 2, continue to the details of the topic in most likely Chapter 5 or Chapter 6, and then look up its evaluations in Chapter 7. Of course, the default ordering of the content refers to

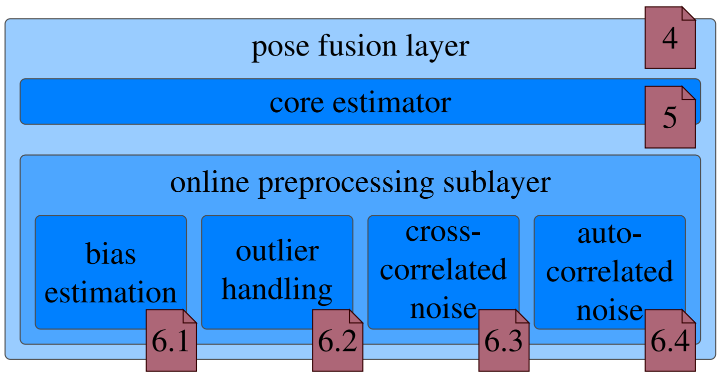


Figure 1.3.: Guidance on where to find the key parts of this thesis. The layer architecture of the pose fusion is detailed in Chapter 4. The core estimator is presented in Chapter 5, and the preprocessing modules are part of Chapter 6.

the preferred arrangement to approach it in its entirety.

2. Related work

In this chapter we provide an overview of the related work on the key topics of this thesis. We start with a review of fusion and state estimation techniques in Section 2.1 and Section 2.2. The next sections Section 2.3 to Section 2.6 follow this up by reviewing approaches related to our preprocessing modules.

2.1. Generic pose fusion

In this section we review the related work on generic pose fusion. The fundamental challenge in this field is to correctly model and integrate imperfect data. The imperfection can manifest itself as measurement uncertainty, outliers, conflicting data, correlated errors, or other effects (Khaleghi et al., 2013). Historically, probability theory is the oldest mathematical theory to deal with data uncertainties. Fusion concepts built on this theory are tailored to reduce the uncertainty. Other fusion concepts are based on different theories, and they focus on different aspects of data imperfection, such as ambiguous or vague data in the case of Dempster–Shafer evidence theory (Murphy, 1998) or random sets (Goodman et al., 2013). This work is based on using tools from probability theory and point estimation because our main challenge is the inherent uncertainty of pose estimates.

The ML solution is obtained by solving the corresponding nonlinear least squares (NLLSQ) problem. Many estimation and data fitting problems are commonly formulated as NLLSQ problems. These include SLAM (Dellaert and Kaess, 2006), bundle adjustment (Agarwal et al., 2009; Triggs et al., 2000), and structure from motion (Forsyth and Ponce, 2002). In these cases the estimation problem is solved by finding the ML solution of a set of robot or camera poses and features in the environment. Naturally, these problems share some common ground with the multi-sensor fusion problem approached in this thesis. However, in contrast to the mentioned problems we need to execute our approach online. Also, we are not interested in reconstructing parts of the environment. These two facts allow us to design the estimation problem in a particularly efficient way while still making use of the sound basis of estimation theory present in the areas of optimization-based SLAM, bundle adjustment, and structure from motion.

Sensor fusion can be distinguished into centralized and decentralized (or distributed) approaches. On the one hand, the former offers the advantage of a high degree of accuracy as all information is available during state estimation. Moreover, it can make use of consistent model assumptions as the central sensor fusion contains all relevant modeling knowledge. Furthermore, it does neither double count information nor be uncertain whether all available information has been processed. On the other hand, decentralized sensor fusion approaches distribute the computational load over multiple hardware units which leads to a more homogeneous processing load. Also, processed data leads usually to less bandwidth requirements between the system components (Darms and Winner, 2005). Distributed sensor fusion with an Extended Kalman filter (EKF) for navigation tasks has been proposed by Weiss et al. (2012). Particle filters have been used for sensor fusion applied to distributed surveillance (Rosencrantz et al., 2002). In this work we focus on centralized sensor fusion because we want to make as few assumptions as possible about the sensor, hardware, and network architecture.

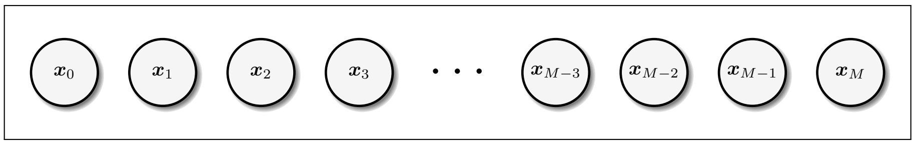
Generic sensor fusion has been applied to other fields than self-localization, too. In the automotive context the concept has been utilized for environment modeling and perception (Munz et al., 2010c,b), mapping (Kubertschak et al., 2014; Grewe et al., 2012), and multi-target tracking (Munz et al., 2010a). These approaches advocate a centralized generic fusion (Munz, 2011) for their specific use cases. Our work is in line with this by proposing a centralized and generic approach for our use case, the pose fusion.

Generic pose fusion is usually motivated by the need for reuse of fusion algorithms. Moore and Stouch (2014) propose a generic pose fusion framework that is based on an EKF. A similar framework, that takes care of handling different timing behaviors of input sources, is proposed by Ratasich et al. (2015). Lynen et al. (2013) propose a generalized pose fusion framework dubbed Multi-Sensor-Fusion EKF. It is based on the IEKF formulation where new states are created by high-frequency IMU readings. Cucci and Matteucci (2013) propose the ROAMFREE framework for multi-sensor pose tracking and sensor calibration. The system is built up on a pose graph. They view input sources as *logical sensors* and treat them like a black box, which is similar to how we view pose sources. Thus, they effectively perform generic sensor fusion for pose tracking. Chiu et al. (2014) model all sensor readings as constraints in a factor graph and propose a selection strategy to choose among the available sensors. This results in a generic state estimation framework that can integrate new sensors through creation of appropriate preprocessing techniques. In addition to these generic sensor fusion approaches Hertzberg et al. (2013) focus on representing poses on a manifold to enable generic sensor fusion algorithms.

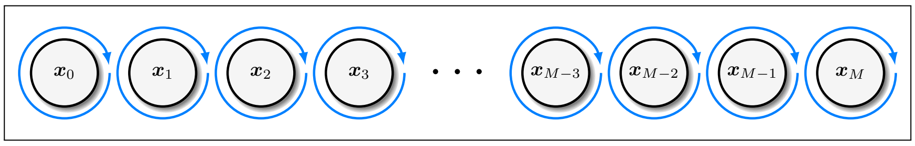
One recurrent insight of the reviewed contributions is that some sort of modularity has to be part of a generic fusion approach. This modularity allows the approach to integrate information from previously unused input sources. We pick up this thought in Chapter 4 where we propose our layered architecture. For now we turn to the related work on the core of the pose fusion. Clearly, any kind of generic pose fusion is based on a concept for state estimation. We therefore review common concepts for this in Section 2.2.

2.2. State estimation for self-localization

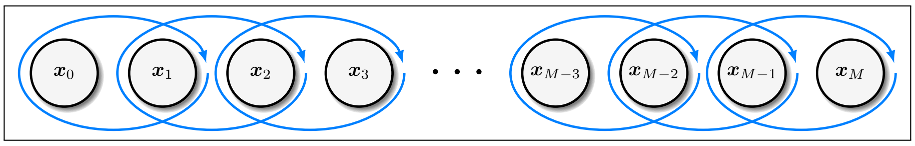
The main task of any pose fusion is to determine the pose of the system. This is achieved by applying techniques from the field of state estimation. For the term *state* we follow the general definition of Simon (2006) who declares that *"the states of a system are those variables that provide a complete representation of the internal condition or status of the system at a given instant of time"*. In this sense, state estimation is *"the problem of reconstructing the underlying state of a system given a sequence of measurements [...]"* (Barfoot, 2017). Applied to our context of self-localization of automated vehicles the narrowest definition of the state is given by the pose of the vehicle in the world reference frame and its uncertainty for a given time. We start by reviewing filtering-based approaches and follow up with a review of smoothing approaches. Generally, the former are only interested in the state at a single instance of time, while the later extend their interest to a sequence of states. Figure 2.1 illustrates the relationship between the EKF, the IEKF, the Sliding Window Filter (SWF), and online (incremental) and offline batch estimation.



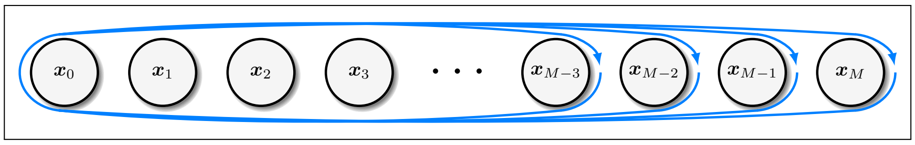
(a) EKF: runs online, but contains only the current state variable and does not iterate.



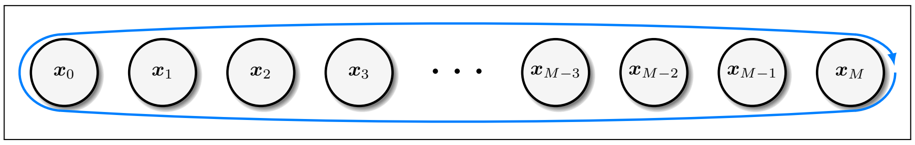
(b) IEKF: runs online and iterates at the current time step, but contains only the current state variable.



(c) Fixed-lag smoothing and SWF: run online and iterate over the set of most recent states variables.



(d) Online batch estimation: iterates over all states up to the current one, but requires more and more runtime with a growing number of state variables.



(e) Offline batch estimation: iterates over all state variables including future ones, but only runs offline.

Figure 2.1.: Comparison of iterative state estimation techniques. The figure is inspired by Barfoot (2017, Fig. 4.17). The arrows indicate the state variables that are being iterated over during one time step.

2.2.1. Filtering-based approaches

Many flavors of filtering-based approaches have been proposed. The Kalman filter is the classical way to estimate the state of a linear system. It consists of a prediction and an update step. The EKF extends it to nonlinear systems, which are common in navigation tasks. It essentially linearizes about the current mean and covariance estimate to apply similar steps as its linear variant.

Many approaches are based upon an EKF formulation. Kubelka et al. (2015) use an error state EKF to fuse positional information from four different odometry sources: an IMU, track encoders, visual odometry, and laser rangefinder scan-matching. Global Positioning System (GPS) is not considered because of its low availability, and a magnetic compass is left out due to its high unreliability in their use cases, leaving the robot with no global position source. The authors describe the modeling of the fusion strategy as a crucial issue mostly complicated by the significantly different update rates ranging from $0.3\,\mathrm{Hz}$ to $90\,\mathrm{Hz}$.

Weiss et al. (2012) propose an EKF to fuse IMU with GPS data and a camera-based pose estimate in the context of micro aerial vehicles. The propagation model is bound to the IMU, making it indispensable while other sensor measurement updates can be integrated in a modular way. The authors detail that integrating measurement delays makes it necessary to recompute all affected states. This is computationally feasible for the state vector but infeasible for the covariance matrix, which they therefore neglect in the case of measurement delays.

Steinhardt and Leinen (2015) propose an error state space EKF that integrates GPS and IMU data with odometry sensors. Emphasis is put on a correction algorithm for measurement latencies for multiple sensors and on modeling the data consistency.

Other filtering-based approaches have been developed to enhance the estimation quality. The IEKF iterates the update and prediction step of the EKF until convergence to minimize the influence of linearization errors. As stated above, Lynen et al. (2013) propose a generic sensor fusion framework built upon the IEKF formulation. The Extended Information Filter (EIF) is the dual of the EKF, where the state belief is parametrized in terms of the information vector and information matrix. It has been extended to the Sparse Extended Information Filter (SEIF) by Thrun et al. (2005). One property of all these approaches is that they represent the state as a unimodal distribution, thus being unable to maintain distinct hypotheses.

The particle filter overcomes this limitation by using a non-parametric representation that permits the approximation of multimodal distributions. Montemerlo et al. (2002) present the FastSLAM algorithm that represents the joint density of the path and the map as a set of particles. Each particle consists of its estimated trajectory and a set of independent Kalman filters for landmarks in the map. Giremus et al. (2004) employ a particle filter to integrate inertial navigation system (INS) and GPS data. The approach described by Mattern et al. (2010) is also based on integrating GPS and odometry data, but additionally augments it by comparing visual and mapped landmarks.

The filtering-based approaches have in common that they rely on the Markov assumption at an early stage and marginalize all older information, thus prematurely incorporating the linearization error. The quality of the linearization is therefore in focus for state estimation. EKFs linearize by computing the first-order Taylor expansion about the current mean estimate and covariance. The Unscented Kalman Filter (UKF) has been developed to provide a better approximation of the nonlinear transformation of the estimated Gaussian random variable, called the *unscented transformation with sigma points* (Julier and Uhlmann, 2004). In contrast, smoothing approaches choose to weaken the influence of linearization errors by frequent relinearization.

2.2.2. Smoothing approaches

Similar to filtering-based approaches there also exist a variety of smoothing approaches. They have in common that they consider the state as a sequence of state variables at specific time instances. If the sequence of these state variables is consecutive and without interruption, then we refer to it as a (discrete) *trajectory*. State estimation for these approaches is based on smoothing the trajectory by applying NLLSQ estimation.

In contrast to filtering techniques, smoothing approaches find a ML estimate by NLLSQ optimization to a Dynamic Bayesian Network (DBN), Markov random field (MRF), or factor graph. Offline batch optimization provides a statistically optimal estimate given additive white Gaussian noise (AWGN). It considers all measurements and optimizes the entire trajectory in a non-causal way. In contrast, online batch estimation considers all information up to the current time step. A drawback of these approaches is that the state vector grows unboundedly over time, thus limiting its online applicability.

This computationally more and more expensive operation becomes feasible through the usage of incremental smoothing techniques. These techniques, such as iSAM2 (Kaess et al., 2012), recompute only the part of the graph that is affected by new measurements. A combination of a long-term smoother using iSAM2 and a short-term smoother using so-called Sliding-Window Factor Graphs is proposed by Chiu et al. (2013) to estimate in parallel the full navigation state and to provide a low-latency state estimate. Their proposal is tightly connected to optimizing a map of landmarks, which pass through three different stages. Indelman et al. (2012) use the incremental smoothing technique proposed by Kaess et al. (2012) to fuse multiple odometry and pose sources. This implies that they choose a similar graph representation as proposed in this contribution with the difference that they keep the full graph in memory over the entire trajectory, making the approach more memory-consuming. Long times of operation, as it is common for vehicles, or memory constraints might lead to issues.

In addition to incremental techniques, smoothing approaches have been adapted to work in an online fashion by restricting the state vector. These are commonly referred to as fixed-lag smoothing algorithms (Maybeck, 1982). As detailed above, filtering-based approaches restrict the state vector to the most recent state, hence collapsing the trajectory estimation into a single pose estimation problem. This, however, prevents relinearization of previous states as they are already marginalized out. Also, the current state is usually not relinearized and the Jacobians are evaluated only once. Strasdat et al. (2012) show that mainly for these two reasons filtering performs suboptimal even for short time frames when compared to NLLSQ estimation.

Fixed-lag smoothing approaches estimate the state over a sliding window of time (Dong-Si and Mourikis, 2011). The lag specifies the size of the sliding window. Often, these approaches are either interested in the state variable at the beginning or the end of the lag. They can be realized with EKFs (as forward-backward smoothing) or optimization-based methods (Ranganathan et al., 2007). They can be seen as an IEKF with an augmented state vector because the filter update of the IEKF is for many problems mathematically identical to the Gauss-Newton method (Bell and Cathey, 1993) when both the prediction and update steps are iterated (Barfoot, 2017).

Keyframe-based approaches are similar to fixed-lag smoothing in that they also maintain a state over a set of state variables. However, the state is not a consecutive sequence of the most recent state variables as in fixed-lag smoothing but instead can contain arbitrarily chosen state variables. These approaches came to the fore in the context of camera-based state estimation (Leutenegger et al., 2015). They commonly use factor graphs to represent the estimation problem.

Sibley (2006, 2007); Sibley et al. (2010) introduce the concept of a SWF in the context of robotics. They apply it to planetary entry, descent, and landing scenarios, in which they estimate surface-structure with a stereo camera setup. Another application is a modified Segway platform that is capable to travel in urban spaces (Newman et al., 2009). Hinzmann et al. (2016) propose a SWF to approach the visual-inertial SLAM problem for fixed-wing unmanned aerial vehicles. The SWF can be considered as a fixed-lag smoothing algorithm for SLAM or structure from motion problems. It extends the state vector to the set of the most recent states variables and estimates the trajectory of the robot and the map of its environment. For this it employs NLLSQ estimation and marginalizes out old state variables.

2.2.3. Marginalization

Smoothing approaches, that estimate the state over a sliding window, have two options to keep the sliding window to a limited size. First, they can simply remove old states and integrate new ones. This is equivalent to conditioning and results in overconfidence. Secondly, they can apply a marginalization strategy that seeks to keep the information of the old states within the optimization problem while removing them from the estimation itself. In the following we review approaches of the second kind that have been applied to pose graphs. In this context node and edge marginalization is applied either exactly or approximately.

Exact marginalization as a fundamental technique to remove parameters from a multivariate Gaussian distribution has been studied extensively in the past. Triggs et al. (2000) gives an introduction with examples from the computer vision and photogrammetry community. They detail the application of the Schur complement as the standard technique for exact marginalization. It presents the challenge of possible creation of fill-in in the system matrix, which is the modification of zero entries to nonzero entries after marginalization. Variable reordering strategies, such as exact (Tinney and Walker, 1967) or approximate minimum degree ordering (Amestoy et al., 1996), try to minimize the fill-in in the system matrix or its Cholesky decomposition. The SWF performs exact node marginalization and Sibley (2006) highlight the effects on different parts of the system matrix.

Marginalization with the help of the Schur complement produces the exact reduced system matrix. The computation though might pose computational challenges, induces fill-in in the system matrix and its Cholesky factorization, and does not provide a semantic representation of the marginalized information.

Approximate marginalization techniques trade exactness for efficiency. The most radical form of approximation is to simply discard nodes and edges. Cucci and Matteucci (2013) delete the oldest nodes from a sliding window without accounting for the removed information. This naturally leads to inconsistent pose and uncertainty estimates but is the fastest possible way of approximate marginalization.

Kretzschmar et al. (2011) thin out pose graphs by removing the nodes which correspond to the least informative laser scans for mapping. They explicitly sparsify the elimination cliques by reducing their number of constraints. To this end, they employ Chow-Liu trees (Chow and Liu, 1968) to locally approximate the Markov blanket of the marginalized nodes. This heuristic seeks to minimize the information loss and to keep the graph sparse.

Vial et al. (2011) propose *Conservative Sparsification* to sparsify the information matrix. This optimization-based method aims to minimize the Kullback-Leibler divergence while enforcing certain edges to be removed, thus resulting in a more sparse matrix. They focus on the effects of edge marginalization instead of node marginalization.

Carlevaris-Bianco et al. (Carlevaris-Bianco and Eustice, 2013; Carlevaris-Bianco et al., 2014) propose to marginalize nodes by replacing them with *generic linear constraints*. These edges are either dense and exact or a sparse Chow-Liu tree approximation. They extend their work (Carlevaris-Bianco and Eustice, 2014) to enforce conservative approximations of the true marginalized potentials.

Huang et al. (2013) derive similar relative constraints from the discarded edges for the remaining nodes. The constraints are conservative, sparse, and account for correlated measurements, thus promoting the consistency of the estimates. The authors propose a ℓ_1-regularized optimization scheme for sparsification instead of using a Chow-Liu tree approximation.

Mazuran et al. (2014) also formulate the sparsification as a convex minimization problem. The optimization only takes into account the Markov blanket of the marginalized node. Additionally, the authors allow for arbitrary nonlinear measurement functions. They extend their work (Mazuran et al., 2016) and embed it in a more general framework called *nonlinear factor recovery*. They show a direct relation between their method and generic linear constraints.

Approximate marginalization techniques seek to avoid the disadvantages of the Schur complement, namely the induced fill-in and the computational burden, by providing a reasonable—if possible conservative—estimate of the influence of the removed parameters. However, this comes at the expense of accuracy. Also, not all approximate approaches are able to relate the removed information to semantically meaningful objects in the model (e.g., nodes and edges).

2.2.4. Timing behavior

In many practical systems the data from different sensors rarely comes perfectly ordered, with a negligible latency, and with comparable and constant frequencies. A specific challenge for state estimation approaches is the integration of out-of-sequence measurements. A conventional filtering-based approach has to propagate its state back to the time of the measurement (retrodiction step), apply it, and apply all stored measurements again. Bar-Shalom (2002) derives an optimal algorithm for a scenario in which a single out-of-sequence measurement has to be applied between the last two updates. The author also notes that an extension to longer time delays involves some kind of non-standard filtering. Steinhardt and Leinen (2015) apply a similar method for an error state space EKF. Challa et al. (2002) propose an augmented state EKF to incorporate out-of-sequence measurements. This can be considered as a filtering-based fixed-lag smoothing solution.

Another approach is to store recent measurements and states, insert the out-of-sequence measurement at the correct place, and recompute all future states (Tessier et al., 2006). This operation is computationally expensive. Therefore, Larsen et al. (1998) propose a fast fusion approach that is suboptimal and only performs well under certain circumstances. Westenberger (2015) shows how to compute retrodiction steps for correlated and non-AWGN noise. The author applies it to an automotive prototype by combining measurements from radio detection and ranging (radar), camera, IMU, and impact sensors.

For graph-based estimation the integration of out-of-sequence or delayed measurements is straightforward. Ranganathan et al. (2007) point out that it is a question of where to add new nodes and edges but that most of the graph structure stays intact. This fundamental difference stems from the fact that optimization-based fixed-lag smoothing algorithms keep a sliding window of state variables and their relations. Out-of-sequence measurements lead to potentially new state variables or new relations that can simply be added to the graph.

2.3. Bias estimation

Measurement and estimation errors can be grouped into *systematic* components, *quasi-stationary* components, and *stochastic* components (Niebuhr and Lindner, 2002). Systematic components are those that are constant for all measurements. Quasi-stationary components are constant for a limited series of measurements. Stochastic components are those that are fully described by a stochastic process. In this thesis we are interested in bias estimation for quasi-stationary errors.

Bias estimation in the context of localization serves to estimate quasi-stationary offsets in pose estimates and to minimize their influence. In the literature it has been primarily treated for GPS-based systems. A common approach consists in augmenting the state vector of a filter to allow for more sophisticated error models and correcting the bias by a second pose source.

Jo et al. (2013) correct quasi-stationary errors of GPS receivers that change slowly over time by comparing visual observations of the road structure to a given road map database. Laneurit et al. (2005) empirically model errors of GPS receivers as an additive Gaussian distribution plus a time-dependent bias and white noise. They estimate the bias by computing the difference of the sensor fusion result to the GPS-based position estimate. Significant bias changes are determined by testing whether the prediction based on the last estimate of the GPS receiver lies within the one sigma error ellipse of the current measurement. Tao et al. (2013) construct a first order autoregressive model for the estimation of the error of a GPS-based system. While this model captures the autocorrelation of the bias, strong bias variations in the form of position jumps are only treated by rejecting the corresponding fixes of the GPS receiver. The authors further compare visual observations of lane markings to a map to correct position errors.

These approaches specifically define a second, unbiased pose estimation to subsequently eliminate the pose bias. Our bias estimation scheme is inspired by the same idea but we generalize it to work with any unbiased pose source.

2.4. Outlier handling

Pose graph optimization is prone to outliers. Constraints that are not well modeled by their Gaussian distribution can lead to huge errors. These errors have quadratic influence in classical NLLSQ and can therefore lead to detrimental effects. As a consequence, robust optimization has been well-studied in the past.

In SLAM problems there are typically three kind of constraints which can potentially lead to outliers. The first one are odometry constraints. Their mean estimate might be inaccurate or the assumption of Gaussian noise might be violated. The second kind are correspondence errors. Many visual SLAM algorithms rely on associating features. This data association is not obvious and can lead to errors. The third kind are loop closure constraints. They are especially important for pure pose graph SLAM problems. However, even few wrong loop closure edges can have disastrous impact. Most research in the SLAM community has focused on handling outliers from wrong loop closure detections as they potentially damage the graph topology severely. This loop closure verification is not directly of interest for a pose fusion. Still, most methods developed for that use case can also be employed for robustification of a pose fusion problem.

Robust optimization is usually based on using robust cost functions. In robust statistics they are commonly called M-estimators. An introduction is given in Section 3.4.3. Choosing the right robust cost function is a difficult task and data set specific. MacTavish and Barfoot (2015) provide a comparison of different robust cost functions for a typical data association problem. Most robust cost functions depend on one or more free parameters. After having found a suitable function these parameters need to be tuned. Agamennoni et al. (2015) present a technique for automatically choosing a cost function and tuning its parameters during optimization.

Switchable Constraints (SC) have been proposed by Sünderhauf and Protzel (2012). The method poses the optimization as a regularization problem. It introduces so-called switch variables s_i for this. They act upon the error variables. Their effect can be seen as scaling of the corresponding information matrices by the value of s_i^2. It is important to note that their values are confined to the range of 0 to 1.

Dynamic Covariance Scaling (DCS) (Agarwal et al., 2013) is an extension to SC. It builds upon the same switch variables. However, the authors propose a closed-form solution to finding the values of the s_i. It mainly depends on the constraint's value. Compared to SC this prevents having to add additional parameters to the optimization problem, thus potentially saving execution time and increasing convergence speed.

Max-mixture constraints (Olson and Agarwal, 2013) are based on the idea that the uncertainty of a constraint can be modeled as Gaussian max-mixture distribution. This adds a second hypothesis to all loop closure constraints. It expresses the probability that this constraint is incorrect. Due to its max-mixture formulation it is easier to embed into the optimization framework than similar sum-of-Gaussian mixtures. This is because the max operator can be pushed through in the negative log likelihood computation. The key effect is that the backend can select during optimization the hypothesis that locally maximizes the likelihood of the corresponding constraint's error function.

Realizing, Reversing, Recovering (RRR) (Latif et al., 2013) is a consensus-based algorithm. It checks for subsets of possible loop closures that are consistent with the topology as defined by the odometry. The subsets are created based on loop closures that occurred in short succession. The method requires an already converged graph before it additionally checks whether possible loop closure edges are consistent with the graph. The consistency check is based on statistical tests. In the end, for each constraint a binary decision is taken whether to keep or to reject it.

Several publications study how these methods compare to each other. Latif et al. (2014) compare DCS, SC, max-mixture constraints, and RRR. The comparison takes into account how parameter tuning intensive these algorithms are. Sünderhauf and Protzel (2013) compare SC, max-mixture constraints, and RRR. They find that max-mixture constraints are simpler to implement and potentially faster while SC seems to need less fine-tuning and performs generally well. Depending on the data set, RRR seems to either outperform the other two methods or struggles to provide a solution at all. Pfeifer et al. (2016) focus more on sensor fusion applications. They simulate non-line-of-sight (NLOS) GPS errors. All of these comparisons have in common that it seems to be difficult to define a clear winner. Depending on the application, the parameter set, and the data set, the results differ.

2.5. Correlated errors between pose sources

Generic pose fusion systems do not know about the specific source of correlation and have to perform *fusion under unknown correlation*. One approach to this is ellipsoidal intersection (Sijs and Lazar, 2012), which maximizes the common information given that the sources share a common prior. Another method is CI (Julier and Uhlmann, 1997), which combines estimates with unknown error correlations. It provides a conservative estimate of the actual mean square error matrix. CI has been employed for a range of applications, including graph-based (Noack et al., 2015) and filtering-based (Julier and Uhlmann, 2007) SLAM. Reinhardt et al. (2012) propose closed-form solutions for two- and three-dimensional matrices on top of which we design our treatment of dependent information. Weighted Geometric Mean (WGM) extends this approach to multiple measurements, see Section 3.5.5.

2.6. Autocorrelated errors

In Section 2.5 we reviewed the related work for fusing measurements with cross-correlated errors. These approaches do not make any assumption about the origin of those correlations and therefore cover all types of correlated errors, including both cross-correlated and autocorrelated errors. Similarly, it is common in SLAM problems to inflate covariance matrices to reflect dependent information (Julier, 2003). However, these approaches are conservative as they implicitly have to assume the worst case of maximum correlation. If we have more information about the nature of the correlation, then we can achieve a more precise fusion result. In this section we dwell on autocorrelated errors. These are of interest to us because we can make use of our knowledge about their autocorrelation.

Intuitively speaking, measurements have autocorrelated errors if the current error depends upon the last errors. Formally, autocorrelated errors can be represented by an autoregressive model as described in Section 3.6. This kind of correlation is also sometimes called *serial correlation* or *temporal correlation*. It is common for time series or filtered data, such as GPS data. Miller et al. (2011) model the GPS position error as an autocorrelated term plus AWGN. They include it in the update step of their particle filter. Other approaches consist in augmenting the model of Kalman filters (Kuhlmann, 2003). In the context of NLLSQ, Cochrane and Orcutt (1949) integrate explicit knowledge about the autocorrelated errors into the estimation problem. This increases the complexity of the problem as it reduces its sparsity. However, it also provides an optimal way of solving least squares regression with variables that contain autocorrelated errors. Building up on this method we derive in Section 6.4 our method that aims to combine both estimation performance with estimation speed.

3. Fundamentals

In this chapter we present fundamental background for the later chapters. We start with giving information about coordinate frames in Section 3.1, different kinds of input sources in Section 3.2, and maps in the automotive context in Section 3.3. After that, we detail in Section 3.4 the algorithmic foundation for the core estimator. This includes an introduction to the corresponding notation and to our view on the optimization technique. It makes up a substantial portion of this chapter as many concepts in this thesis build up on this theory. We complete the chapter with Section 3.5, Section 3.6, and Section 3.7 by presenting important concepts for our preprocessing modules.

3.1. Reference frames

Throughout this thesis we mainly use two reference frames: the vehicle reference frame and a world reference frame. The former is used whenever we want to express a location or an orientation from the point of view of a certain pose of the vehicle. This could be the distance traveled between two time steps, for example. If only the vehicle's relative movement is of interest, then this is independent of its pose with respect to the world. The latter is used to express where a vehicle is located in the world. This is useful if we want to look up information in a map, for example.

The following two subsections detail these two reference frames.

3.1.1. Vehicle reference frame

In this thesis the vehicle reference frame is a reference frame defined according to ISO 8855 (International Organization for Standardization, 2011). It is also commonly referred to as body frame or body coordinate system (Haken, 2013). The mobile robotics community uses for the most part the same conventions (see Grisetti et al. (2010a) and

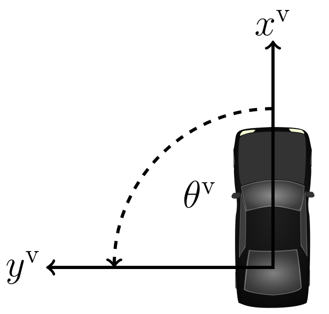


Figure 3.1.: Vehicle reference frame. θ_{v} is zero on the x-axis and increases to the left. The z-axis (not shown) is pointing upwards. This reference frame is useful for describing headings towards and positions of objects relative to the current position of the vehicle.

Kelly (2013) for two examples). Figure 3.1 shows the definition of this reference frame. Its origin lies at the projection of the rear axis' center on the ground. The x-axis is always parallel to the support surface. It points from the origin to the front of the vehicle. We use the superscript v to indicate variables defined in the vehicle reference frame. As an example, $\boldsymbol{z}_i^{\mathrm{v}}$ could denote the i-th measurement vector in this frame.

3.1.2. World reference frame

The Universal Transverse Mercator (UTM) coordinate system is based on a conformal cylindrical map projection. A map projection is conformal if the projection of lines, which intersect on the ellipsoid, intersect at the same angle. The UTM system divides the earth into sixty zones, each 6° of longitude in width (up to $800\,\mathrm{km}$), and defines a two-dimensional Cartesian coordinate systems within each zone. For each zone a different secant transverse Mercator projection (see Figure 3.2) is used to obtain the Cartesian grid. This is necessary as directions, distances, and areas are only reasonably accurate within a few degrees of the central meridian.

The x-axis faces east while the y-axis faces north. Positions on the axes are measured in meters. Additionally, we define the global heading to be zero radian at east with increasing values towards north. It is bound between $[0, 2\pi)$.

We use the UTM system throughout this thesis as the world reference frame because we want to adapt a standardized two-dimensional Cartesian reference frame with low distortions of distances. It could be interchanged with another reference frame without

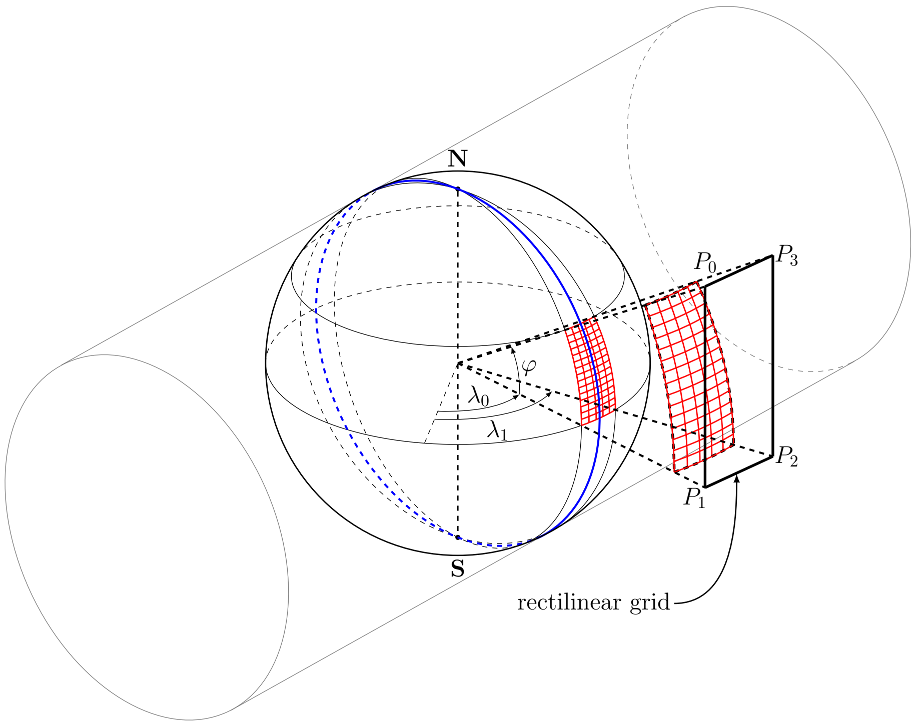


Figure 3.2.: A secant transverse Mercator projection. **N** and **S** mark the north and south pole. λ_0 and λ_1 are the longitudes at the border of the grid. In the UTM system the difference between λ_1 and λ_0 is always equal to $6°$. The central meridian is shown in blue. The projection of the red grid onto the cylinder results in the black rectilinear grid with the end points P_0, P_1, P_2, and P_3. It is a conformal map of the sphere's surface covering the area between the equator and the latitude φ in the longitude band between λ_0 and λ_1. The figure is adapted from Miani (2009).

affecting the contributions of this thesis as the methods developed in this thesis do not depend on it. A local tangent plane as used by an East North Up (ENU) coordinate system could be used alternatively, for example. We use the superscript w to indicate variables defined in the world reference frame. As an example, $\boldsymbol{z}_i^{\mathrm{w}}$ could denote the i-th measurement vector in the world reference frame.

3.2. Self-localization of automated vehicles

In Section 3.1 we detailed the global and local coordinate frames that accompany us throughout the rest of the thesis. In this section we survey different self-localization techniques that estimate poses in these coordinate frames. This highlights the multitude of today's available self-localization methods and emphasizes the need of a pose fusion.

There are many ways to gain information about the pose of the vehicle. We cluster these methods into those that provide globally referenced information and those that provide relative movement information. We refer to pose sources which estimate poses in the world reference frame as *global pose sources* and to poses in this system as *global poses*. Pose sources which measure spatial transformations relative to the previous pose are dubbed *local pose sources* or simply odometry.

Odometry and global pose sources have orthogonal strengths and weaknesses. On the one hand, odometry measurements are usually available at high frequencies and do not require a priori knowledge about the environment. On the other hand, they accumulate drift with growing distance and they are not globally referenced. The properties of global pose measurements are typically converse to this. On the upside, they are globally referenced and their error is independent of the covered distance. On the downside, they are usually only available at low frequency and require a priori knowledge about or preparation of the environment, such as maps or satellite placement. Combining both types of measurements in our pose fusion allows us to estimate a trajectory which is both globally referenced and locally smooth.

In Section 3.2.1 and Section 3.2.2 we highlight some of the most common global pose and odometry sources for robots and automated vehicles. This demonstrates that there are many potential localization systems that can serve as input for our pose fusion approach. In Section 7.2 we provide information about pose sources that we use in the experiments.

3.2.1. Global pose sources

Many systems are able to estimate poses in a global reference frame. In the automotive context the presumably most known and most employed systems are GNSS-based, such as GPS, Global Navigation Satellite System (GLONASS), Galileo, or BeiDou Navigation Satellite System (BDS). The corresponding receivers all have in common that they determine their location based on the radio signals emitted by satellites orbiting the earth. Systems that are based on measuring the Received Signal Strength Indication (RSSI) values of cellular signals such as Global System for Mobile Communications (GSM) (Xue et al., 2016) provide roughly the same accuracy of a couple of meters. However, they are significantly less spread. For decades, satellite-based systems have been the cornerstone of automotive navigation systems.

GNSS-based systems for automotive applications are usually complemented with an IMU such that the system can still provide position estimates even in the absence of satellite signals (e.g., in tunnels). This integration of GNSS/IMU also allows the system to provide information about the orientation, whereas conventional GNSS systems can directly only measure information about the position. Moreover, the coupling enables higher output frequencies of the estimation process (Dudek and Jenkin, 2016). On the upside, GNSS-based systems are widespread and can provide global pose estimates on huge parts of the globe without requiring any kind of map. However, their downsides include that they can be particularly noisy in urban settings or even unavailable due to so-called urban canyons (tall buildings on both sides of the street). There are three main reasons for noisy GPS measurements in these scenarios. The first reason is the constellation of satellites with respect to the receiver, where the radio signals of several satellites may be unobservable due to buildings or other obstructions. This satellite shadowing lowers the number of visible satellites which generally means a lower accuracy in the positioning solution. Also, if the field of view towards the satellites is limited in such a way that only transmissions from satellites in a narrow field of view are available, then the geometry of the relative positions between satellite and receiver is potentially degraded. The second reason are multipath effects where the same satellite transmission arrives at the receiver's antenna over multiple paths (e.g., due to reflections on buildings or walls). This causes the receiver to observe multiple times the same transmission with different delays. The third reason are NLOS propagation and diffraction of satellite signals where the direct signal is blocked and only a single reflected or diffracted signal is received. This differs from multipath effects, in which the direct signal is usually one of the received signals. Among these three different error sources, Zimmermann et al. (2016) suggest that a degraded satellite geometry is less influential than multipath and NLOS effects. Several techniques for mitigating the effects of multipath and NLOS reception exist. These can generally be classified into hardware-based (e.g., antenna design, choke rings), receiver-based, and post-receiver-based techniques. Groves et al. (2013) provides an overview and proposes a portfolio approach to multipath and NLOS mitigation.

With the advent of advanced driver assistance systems and automated driving functions arose the demand of a higher level of positioning accuracy than what conventional satellite-based automotive navigation systems can provide. This lead to the use of more sophisticated GNSS-based concepts in research vehicles. These include differential positioning techniques such as Differential Global Positioning System (DGPS), Real Time Kinematics (RTK), Precise Point Positioning (PPP), and combinations thereof (Urmson et al., 2008; Kammel et al., 2008; Klingbeil et al., 2014; Eling et al., 2015; Schneider et al., 2016a; Eling, 2016). While these techniques promise up to centimeter-level precision under good conditions, it remains unclear whether these systems can provide sufficient accuracy under all relevant driving conditions, including tunnels, underground parkings, urban canyons etc. Also, their high costs make their entrance into the mass market difficult in the near future.

For these reasons, pose sources based on different, potentially cheaper sensors came to the fore. In this context, the wording *cheaper* also includes sensors that have been originally installed for other ADAS functions and can be additionally exploited for localization purposes without additional cost. As GNSS-based systems rely on measuring signals from satellites, they do not require a map. Conversely, global localization systems for robotic and automotive driving applications based on most other sensors typically compare their current sensor data to previously mapped information. In this sense, a map is defined by providing geographically referenced information. If this map is expressed in a global coordinate frame, then the corresponding localization system can be treated as a global pose source.

There are different ways to obtain maps that are useful for localization. For some applications, it can be beneficial to reuse existing map material. Map-based localization approaches of this kind include using publicly available maps such as OpenStreetMap (Floros et al., 2013) or Google Street View (Agarwal et al., 2015). An alternative is to create maps by “mapping with known poses” (Thrun, 2002; Stachniss, 2006; Merfels, 2014), which involves some form of reference positioning technique. The strong assumption that all poses are perfectly known, if justified, substantially eases the map building process. In the domain of mobile robotics, many state-of-the-art localization systems are based on SLAM approaches (Stachniss et al., 2016). Durrant-Whyte and Bailey (2006) as well as Bailey and Durrant-Whyte (2006) provide a survey of the rather early developments of SLAM, while Huang and Dissanayake (2016) provide a more recent critique. The SLAM problem is essentially the problem of mapping an unknown environment and at the same time using this map to determine the robot's position. Maps and localization results by SLAM approaches can exist in a global coordinate frame, such as UTM, if they incorporate for example GPS constraints.

Many different sensors are utilized for map-based localization or to solve the SLAM problem. Among these, image- and video-based approaches are prevalent due to the price and ubiquity of cameras. Different types of cameras have been used for the localization task, including monocular cameras (Lategahn et al., 2013; Heidenreich et al., 2015), stereo cameras (Ziegler et al., 2014), and multi-camera fisheye setups (Houben et al., 2015). Laser sensors are another prominent example of sensors that are typically utilized for localization. They are the backbone of many localization techniques in the form of lidar sensors (Levinson et al., 2007; Vysotska and Stachniss, 2016; Schlichting and Brenner, 2014). They measure the distances to objects by illuminating them with laser light (active sensing) and measuring the reflections. Another sensor type with a similar mode of operation is radar, which emits radio waves to measure the angles, velocities, and distances of objects. By comparing the radar measurements to previously mapped information of stationary objects, one can determine its own location (Werber et al., 2015; Ward and Folkesson, 2016). More exotic sensors, such as radio-frequency identification (RFID) readers, have also been utilized for vehicle self-localization (Qin et al., 2017).

Often, cameras, laser scanners, or radar serve as the main sensor for a localization approach, but they are rarely the sole one. Instead, they are commonly being combined with other information, e.g., GPS and IMU data. Some approaches employ other sensor combinations, such as camera and lidar (Wolcott and Eustice, 2014; Caselitz et al., 2016), to determine the pose of the robot across sensor modalities. The bottom line is, there are many different localization approaches that share the common ground that they act as global pose sources.

3.2.2. Odometry sources

Odometry is the estimation of the relative trajectory of the own vehicle (Dudek and Jenkin, 2016). Different sensors can be used to compute odometry information. Among these, IMUs are prevalent in the automotive industry. They typically consist of a combination of accelerometers and gyroscopes. The former measures external forces on the vehicle while the latter measures local changes in rotation. Both are commonly part of the electronic stability program (ESP). These sensors are sometimes combined with wheel speed and steering wheel angle sensors to increase the accuracy and reliability. An IMU is typically the main sensor for an INS. The main task of an INS is to perform *dead reckoning* by continuously integrating the measured accelerations and velocities to compute a position fix. This is locally smooth but not globally referenced. Furthermore, it accumulates drift over time because small measurement errors are accumulated over time and are never corrected. In this thesis we will use such an INS which is dubbed *EgoMaster* and is the successor to the system described by Baer et al. (2009). We give additional technical details in Section 7.2.1.

Cameras are another source for odometry information. Monocular cameras as well as stereo cameras have been successfully used to estimate the vehicle's egomotion (Scaramuzza and Fraundorfer, 2011; Fraundorfer and Scaramuzza, 2012). The corresponding process is called visual odometry (Nistér et al., 2005). It is based on computing the egomotion on image features (Cvišić and Petrović, 2015; Schneider et al., 2016b) using sparse feature-based, semi-dense (Engel et al., 2013; Forster et al., 2014), or dense matching methods (Lovegrove et al., 2011).

Lidar and radar can also prove useful for odometry estimation. Approaches of this kind typically compute odometry estimates by scan matching of laser (Olson, 2009) or radar (Rapp et al., 2015) measurements. Moreover, some approaches have successfully exploited combinations of the aforementioned sensors, e.g., vision and lidar (Zhang and Singh, 2015), vision and inertial (Wang et al., 2013)).

3.3. Maps for automated driving

Automated driving relies on suitable maps. These are available with different content and different formats. Section 3.3.1 briefly describes the current industry standard, while Section 3.3.2 highlights the map format that is being used at Volkswagen Group for research and predevelopment purposes. In Section 6.2 we make use of the fact that we can extract the center line from these maps. Therefore, we explain this concept and its context here.

3.3.1. Navigation Data Standard

The Navigation Data Standard (NDS) association[1] is a group of car manufacturers, map suppliers, and infotainment system suppliers. Their main goal is to standardize the physical storage format for map data to achieve exchangeability, compatibility, and interoperability of navigation databases between different suppliers and systems.

The standard describes a binary format and how to compile it. It is designed for efficient retrieval of content while minimizing storage space. It also describes update mechanisms for parts or the entirety of the data and provides a digital rights management. The data is organized per use case into so-called building blocks like routing, search, or points of interest.

Roads are represented by links which form the topological road network. Roads with multiple lanes can be combined into lane groups which are logically connected to links. For each lane, its geometrical shape and the center line are defined with georeferenced coordinates. Additional attributes such as speed limits can be connected to links and lanes as well.

3.3.2. Detailed Lane Model

The Detailed Lane Model (DLM) is an extension to the NDS. Whereas NDS is maintained and developed by an industry-wide association of members and designed for series products, the DLM is developed by Volkswagen Group for internal use and provides features that are not (yet) standardized in NDS. It is meant as a format for research with the flexibility to add or change elements as needed. The DLM is based on and extends the NDS maps. If certain developments of the DLM are found to be useful as possible future NDS elements, they are brought into the standardization groups for further discussion.

[1]`http://www.nds-association.org/`

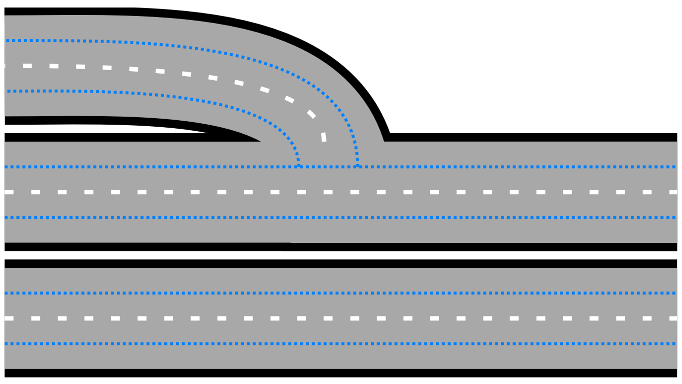


Figure 3.3.: Section of a DLM that has three roads with two lanes each. Each road has a different road boundary (black lines). Additionally, each lane has a different center line (blue dashed). The lane boundary between adjacent lanes is depicted with white dashed lines.

Within the context of this thesis, DLM maps are the only maps that are of interest to us. We are particularly interested in the lane geometry information. Figure 3.3 shows three roads with their respective road boundaries and center lines. They describe the precise geometry with georeferenced line strings. A line string is a piecewise linear curve that is also sometimes called a polyline. It is simply a connected series of line segments where the line segments are defined by start and end points.

The concept of lane groups in NDS maps is represented by DLM segments. Figure 3.4 shows how a section of a DLM is partitioned into multiple segments. Adjacent lanes with the same direction of travel are grouped to segments such that attributes of a segment apply to all of its elements. Figure 3.5 shows a section of a real DLM.

In this thesis, maps are of interest in two places. First, we make use of the lane geometry information in Section 6.2. Secondly, we occasionally use visualizations overlaid on a DLM to give a better understanding of the underlying road structure.

3.4. Nonlinear least squares on manifolds

In this thesis we adopt an optimization-based view on the problem of pose fusion. To this end, we provide in this chapter a brief self-contained overview of the NLLSQ method that is the foundation of our pose fusion.

We begin in Section 3.4.1 with a brief statement of the problem that NLLSQ methods

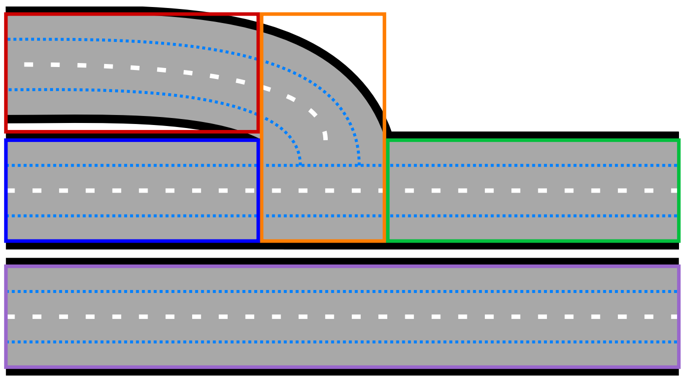


Figure 3.4.: DLM segments, differentiated by color. A new segment starts whenever the topology of the road changes.

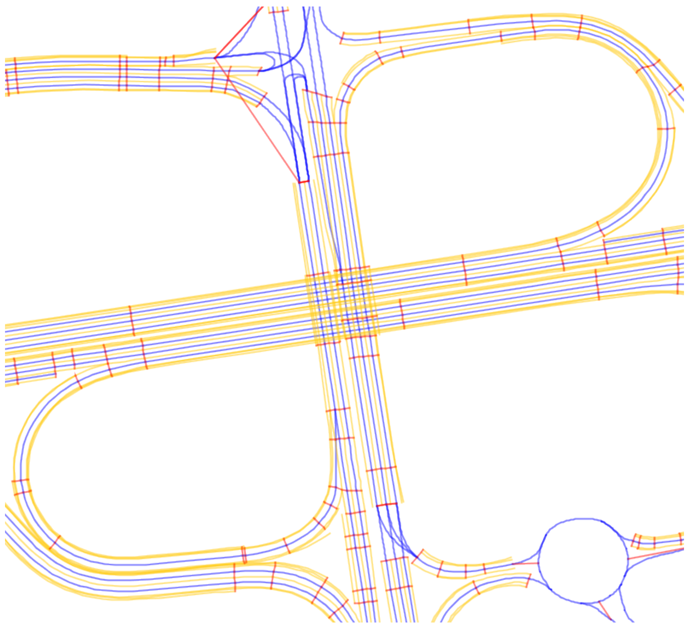


Figure 3.5.: Section of a real DLM around Wolfsburg. The blue lines represent the center lines while the yellow lines visualize the lane boundaries. The segment boundaries are shown in red.

seek to solve. With this formal problem definition at hand, we present two optimization methods in Section 3.4.2 to approximately solve it, namely the Gauss-Newton and the Levenberg-Marquardt algorithms. In Section 3.4.4 we extend the optimization approach from Euclidean vector spaces to manifolds. Section 3.4.5 discusses the probabilistic interpretation of this optimization approach. In Section 3.4.6 we detail a graph-based representation of the optimization problem, which we use frequently throughout this thesis. As the wording and notation between the geodetic and the robotics community are sometimes different we bridge this gap by explicitly detailing the relation of the notations and wordings in Section 3.4.7.

3.4.1. Problem statement

Least squares methods are a family of approaches designed to approximately solve overdetermined equation systems. They all aim to minimize the sum of squared errors of the computed solution. Many estimation and data fitting problems can be formulated as linear or nonlinear least squares problems. NLLSQ methods are used to approximate solutions in case that the residuals are not linear in all unknowns.

Let $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_m^\top\right]^\top$ be the state vector, $\boldsymbol{z}_i$ be a measurement of the i-th state variable $\boldsymbol{x}_i$, and $\boldsymbol{h}_i$ be a (nonlinear) function that maps $\boldsymbol{x}$ to a predicted measurement $\boldsymbol{h}_i(\boldsymbol{x})$. Given a set of n noisy measurements $\boldsymbol{z} = \{\boldsymbol{z}_i\}_{i=1}^n$, the problem consists in estimating the state $\boldsymbol{x}^*$ which best explains the measurements $\boldsymbol{z}$.

This explanation is defined to be the state vector $\boldsymbol{x}^*$ that yields the smallest global error. The global error function is the sum of the squared individual error terms. The individual error terms $\boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i)$ are defined as the difference between the actual and the predicted measurements with

$$\boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i) = \boldsymbol{h}_i(\boldsymbol{x}) - \boldsymbol{z}_i. \tag{3.1}$$

The individual squared error terms (also often called residuals) are the scalars

$$e_i(\boldsymbol{x}, \boldsymbol{z}_i) = \boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i)^\top \boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i). \tag{3.2}$$

If additional information about the reliability of the observations is available, it can be incorporated by weighting the error terms accordingly such that

$$e_i(\boldsymbol{x}, \boldsymbol{z}_i) = \boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i)^\top \boldsymbol{\Lambda}_i \boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i), \tag{3.3}$$

where $\boldsymbol{\Lambda}_i$ is typically chosen as the inverse of the measurement's covariance matrix (see Section 3.4.5). Depending on the context, $\boldsymbol{\Lambda}_i$ is commonly called *information matrix*, *precision matrix*, or *weighting matrix*.

The global error function $F(\boldsymbol{x})$ is defined as

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} e_i(\boldsymbol{x}, \boldsymbol{z}_i). \tag{3.4}$$

The goal of the NLLSQ approach is then formalized as finding the state $\boldsymbol{x}^*$ which minimizes the global error function such that

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}). \tag{3.5}$$

3.4.2. Optimization of nonlinear least squares problems

An analytical solution for the optimization problem in (3.5) can be found by deriving the global error function and finding its roots. This is in general complex—if not impossible—as there is often no simple closed-form solution. We therefore resort to numerical approaches.

Gauss-Newton algorithm

The Gauss-Newton algorithm is a numerical approach and approximates the solution of (3.5). It consists of choosing an initial state $\breve{\boldsymbol{x}}$, linearizing the individual error terms around $\breve{\boldsymbol{x}}$, and solving a system of linear equations. These steps are iterated until convergence.

The individual error terms are linearized by approximation to a first-order Taylor series expansion about $\breve{\boldsymbol{x}}$. Therefore, we analyze the value of the global error function at the initial state $\breve{\boldsymbol{x}}$ plus an update vector $\Delta\boldsymbol{x}$:

$$F(\breve{\boldsymbol{x}}+\Delta\boldsymbol{x}) = \sum_{i=1}^{n} e_i(\breve{\boldsymbol{x}}+\Delta\boldsymbol{x},\boldsymbol{z}_i) \tag{3.6}$$

$$= \sum_{i=1}^{n} \boldsymbol{e}_i(\breve{\boldsymbol{x}}+\Delta\boldsymbol{x},\boldsymbol{z}_i)^\top \boldsymbol{\Lambda}_i \boldsymbol{e}_i(\breve{\boldsymbol{x}}+\Delta\boldsymbol{x},\boldsymbol{z}_i). \tag{3.7}$$

It is useful to compute the linearization of $\boldsymbol{e}_i(\breve{\boldsymbol{x}}+\Delta\boldsymbol{x},\boldsymbol{z}_i)$ as

$$\boldsymbol{e}_i(\breve{\boldsymbol{x}}+\Delta\boldsymbol{x},\boldsymbol{z}_i) \approx \boldsymbol{e}_i(\breve{\boldsymbol{x}},\boldsymbol{z}_i) + \breve{\boldsymbol{J}}_i\Delta\boldsymbol{x}, \tag{3.8}$$

$$\breve{\boldsymbol{J}}_i = \left.\frac{\partial \boldsymbol{e}_i(\boldsymbol{x},\boldsymbol{z}_i)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}}, \tag{3.9}$$

where we neglect the higher order terms in (3.8) by assuming that the increment $\Delta\boldsymbol{x}$ is sufficiently small.

Plugging (3.8) into (3.7) and writing $\breve{\boldsymbol{e}}_i = \boldsymbol{e}_i(\breve{\boldsymbol{x}})$, we obtain the linearized global error function

$$F_{\text{lin}}(\Delta\boldsymbol{x}) = \sum_{i=1}^{n} (\breve{\boldsymbol{e}}_i + \breve{\boldsymbol{J}}_i\Delta\boldsymbol{x})^\top \boldsymbol{\Lambda}_i (\breve{\boldsymbol{e}}_i + \breve{\boldsymbol{J}}_i\Delta\boldsymbol{x}) \tag{3.10}$$

$$= \sum_{i=1}^{n} \underbrace{\breve{\boldsymbol{e}}_i^\top \boldsymbol{\Lambda}_i \breve{\boldsymbol{e}}_i}_{c_i} + 2\underbrace{\breve{\boldsymbol{e}}_i^\top \boldsymbol{\Lambda}_i \breve{\boldsymbol{J}}_i}_{\boldsymbol{b}_i^\top} \Delta\boldsymbol{x} + \Delta\boldsymbol{x}^\top \underbrace{\breve{\boldsymbol{J}}_i^\top \boldsymbol{\Lambda}_i \breve{\boldsymbol{J}}_i}_{\boldsymbol{H}_i} \Delta\boldsymbol{x} \tag{3.11}$$

$$= \sum_{i=1}^{n} c_i + 2\boldsymbol{b}_i^\top \Delta\boldsymbol{x} + \Delta\boldsymbol{x}^\top \boldsymbol{H}_i \Delta\boldsymbol{x} \tag{3.12}$$

$$= \underbrace{\sum_{i=1}^{n} c_i}_{c} + 2\underbrace{\sum_{i=1}^{n} (\boldsymbol{b}_i^\top)}_{\boldsymbol{b}^\top} \Delta\boldsymbol{x} + \Delta\boldsymbol{x}^\top \underbrace{\sum_{i=1}^{n} (\boldsymbol{H}_i)}_{\boldsymbol{H}} \Delta\boldsymbol{x} \tag{3.13}$$

$$= c + 2\boldsymbol{b}^\top \Delta\boldsymbol{x} + \Delta\boldsymbol{x}^\top \boldsymbol{H} \Delta\boldsymbol{x}. \tag{3.14}$$

All in all, the global error function is approximated via a quadratic form in the variable $\Delta\boldsymbol{x}$ to

$$F(\breve{\boldsymbol{x}}+\Delta\boldsymbol{x}) \approx F_{\text{lin}}(\Delta\boldsymbol{x}) = c + 2\boldsymbol{b}^\top \Delta\boldsymbol{x} + \Delta\boldsymbol{x}^\top \boldsymbol{H} \Delta\boldsymbol{x} \tag{3.15}$$

with

$$c = \sum_{i=1}^{n} \breve{\boldsymbol{e}}_i^\top \boldsymbol{\Lambda}_i \breve{\boldsymbol{e}}_i \tag{3.16}$$

$$\boldsymbol{b}^\top = \sum_{i=1}^{n} \breve{\boldsymbol{e}}_i^\top \boldsymbol{\Lambda}_i \breve{\boldsymbol{J}}_i \tag{3.17}$$

$$\boldsymbol{H} = \sum_{i=1}^{n} \breve{\boldsymbol{J}}_i^\top \boldsymbol{\Lambda}_i \breve{\boldsymbol{J}}_i. \tag{3.18}$$

With this linearized global error function our goal is to find the optimal update vector

$$\Delta\boldsymbol{x}^* = \arg\min_{\Delta\boldsymbol{x}} F_{\text{lin}}(\Delta\boldsymbol{x}). \tag{3.19}$$

As $F_{\text{lin}}(\Delta\boldsymbol{x})$ is of quadratic form it is straightforward to find its minimum value by deriving it with respect to $\Delta\boldsymbol{x}$, setting the first derivative to zero, and inspecting the second derivative. The first derivative is

$$\frac{\partial F_{\text{lin}}(\Delta\boldsymbol{x})}{\partial \Delta\boldsymbol{x}} = 2\boldsymbol{b} + (\boldsymbol{H} + \boldsymbol{H}^\top)\Delta\boldsymbol{x} \tag{3.20}$$

$$= 2\boldsymbol{b} + 2\boldsymbol{H}\Delta\boldsymbol{x}. \tag{3.21}$$

The minimum value of $F_{\text{lin}}(\Delta\boldsymbol{x})$ is obtained by setting this derivative to zero, which leads to

$$\boldsymbol{0} = 2\boldsymbol{b} + 2\boldsymbol{H}\Delta\boldsymbol{x}^* \tag{3.22}$$

$$\Leftrightarrow \boldsymbol{H}\Delta\boldsymbol{x}^* = -\boldsymbol{b}. \tag{3.23}$$

Inspecting the second derivative shows that it does not depend on $\Delta\boldsymbol{x}$ as

$$\frac{\partial^2 F_{\text{lin}}(\Delta\boldsymbol{x})}{\partial (\Delta\boldsymbol{x})^2} = 2\boldsymbol{H}. \tag{3.24}$$

As $\boldsymbol{H}$ is symmetric and positive semi-definite we see that $\Delta\boldsymbol{x}^*$ is a minimum of $F_{\text{lin}}(\Delta\boldsymbol{x})$.

The optimal update vector is therefore computed by solving the system of linear equations given in (3.23). In theory, the solution is readily obtained by left multiplying with $\boldsymbol{H}^{-1}$. In practice, this computationally expensive operation (matrix inversion) can be averted by solving (3.23) with Cholesky decomposition (as $\boldsymbol{H}$ is a symmetric positive semi-definitive matrix), QR factorization, or using an iterative method such as conjugate gradient (in case that $\boldsymbol{H}$ is sparse).

Solving (3.19) allows us to compute (3.5) by adding the optimal update vector to the initial guess:

$$\boldsymbol{x}^* = \breve{\boldsymbol{x}} + \Delta\boldsymbol{x}^*. \tag{3.25}$$

This solution is optimal given the linearization point $\breve{\boldsymbol{x}}$. Due to the linearization, however, we need to iterate this procedure until some convergence criterion is met. This is because we introduced approximation errors by neglecting the higher order terms of the first-order Taylor series expansion of $\boldsymbol{h}_i(\boldsymbol{x})$ in (3.8). In each iteration we first update the linearization point $\breve{\boldsymbol{x}}$ to the solution $\boldsymbol{x}^*$ of the last iteration as given in (3.25), recompute the linearization as given in (3.15), and then solve for $\Delta\boldsymbol{x}^*$ as given in (3.23).

This iterative procedure assumes that the error functions behave smoothly in the neighborhood of the minimum. It is however not guaranteed that the minimum is a global minimum. Different techniques exist to deal with local minima, for example choosing multiple different initial values such that the global minimum is among the set of computed minima.

Levenberg-Marquardt algorithm

In its direct implementation the Gauss-Newton algorithm will not always yield a decrease in every iteration for $F(\boldsymbol{x})$. This occurs when the update vector $\Delta\boldsymbol{x}^*$ overshoots the optimal estimate. One strategy to handle this kind of divergence is to add only a fraction α of the update vector to the linearization point so that (3.25) changes to

$$\boldsymbol{x}^* = \breve{\boldsymbol{x}} + \alpha\Delta\boldsymbol{x}^* \tag{3.26}$$

with $0 < \alpha < 1$.

A more sophisticated approach of treating this kind of divergence is the Levenberg-Marquardt algorithm. It consists in replacing (3.23) with

$$(\boldsymbol{H} + \lambda\boldsymbol{I})\Delta\boldsymbol{x}^* = -\boldsymbol{b}. \tag{3.27}$$

This introduces the damping parameter λ, which is typically adapted throughout the optimization iterations. Its purpose is to robustify the Gauss-Newton algorithm by being less susceptible to bad initial guesses at the expense of a slightly slower convergence rate.

3.4.3. Robustified least squares

The presented NLLSQ methodology can be applied to many estimation problems. In this basic formulation, however, it is known to be sensitive to outliers. Therefore, in many real-world applications it is necessary to robustify the estimation scheme against the influence of outliers. A common approach for this are so-called *robust cost functions* (Förstner and Wrobel, 2016).

The main influence of outliers comes from the quadratic characteristics of the cost function. An outlier causes a large error value that the optimization seeks to minimize at the cost of falsely accepting increased residuals for the non-outlier measurements. Therefore, the general idea of robust cost functions is to adapt this function such that large error values are weighted less than quadratically. Specifically, instead of seeking the minimum of the global error function (see (3.4)), we instead seek the minimum of the robustified global error function

$$F_{\text{rob}}(\boldsymbol{x}) = \sum_{i=1}^{n} \rho_i(e_i(\boldsymbol{x}, \boldsymbol{z}_i)), \tag{3.28}$$

where $\rho_i(e)$ is a robust cost function. While different choices for this function are possible a common choice is the Pseudo-Huber cost function (Hartley and Zisserman, 2004). It is defined by

$$\rho_{\text{H}}(e) = 2\delta^2 \left(\sqrt{\frac{e}{\delta^2} + 1} - 1 \right), \tag{3.29}$$

where δ is a free parameter. The intuition behind this function is to scale error terms quadratically in the local vicinity of $e = 0$ and linearly for large values. Note that we can apply a different cost function for each constraint. For a derivation of how the robust cost function changes the corresponding matrix $\boldsymbol{H}$ and the vector $\boldsymbol{b}$, we refer the reader to the work of Triggs et al. (2000). We review additional techniques for robustified least squares in Section 2.4.

3.4.4. Optimization on smooth manifolds

The Gauss-Newton algorithm as presented in its standard form in Section 3.4.2 is designed for Euclidean vector spaces. In case that estimation problems occur outside such spaces it is often possible to generalize the solution on smooth manifolds. An example are estimations in the special orthogonal group $SO(2)$ which arise when estimating an orientation in the plane. The orientation is described by a single angle $\alpha \in [-\pi, \pi)$. Small, local perturbations around α can be treated with the canonical tool set of the $+$ and $-$ operators in the Euclidean plane. However, angular differences greater than π pose a problem. While the difference between $\alpha_0 = -\pi$ and $\alpha_1 = \pi - \epsilon$ is ϵ, naively computing the difference yields $\alpha_0 - \alpha_1 = -2\pi + \epsilon$, which is clearly wrong.

This example illustrates an issue that is more severe for rotations in three-dimensional space. The estimation can neither be carried out on parameters of a minimal representations (e.g., Euler angles) as they suffer from singularities nor of overparametrizations (e.g., rotation matrices or quaternions) as the additional degrees of freedom are not modeled (Hertzberg, 2008).

Embedding the error function on a manifold

The error minimization is applied on a manifold instead of an Euclidean space. A manifold is a topological space that considers the local neighborhood of each point as a Euclidean space, while the space on a global scale is not necessarily Euclidean (Blanco, 2010). An intuitive example is given by a unit sphere (a manifold of dimension two) which looks flat in a sufficiently small surface area while globally being curved. It does therefore not come as a surprise that this parametrization was first developed for geodesic applications (Gabay, 1982).

The key idea is to use an overparametrization for the state variable $\boldsymbol{x}$ and to use a minimal parametrization for a local increment $\Delta\boldsymbol{x}$. Considering the underlying space as a manifold we define two nonlinear operators $\boxplus$ and $\boxminus$ that map a local variation in the Euclidean space to an increment on the manifold (Hertzberg, 2008; Frese and Schröder, 2007; Grisetti et al., 2010b). Figure 3.6 illustrates the effect of the $\boxplus$-operator. Kümmerle (2013) gives possible definitions for the implementation of these operators for 2D and 3D pose graphs. We encapsulate the operations on the manifold by changing the state variable only via these two operators (Hertzberg et al., 2013). Therefore, we

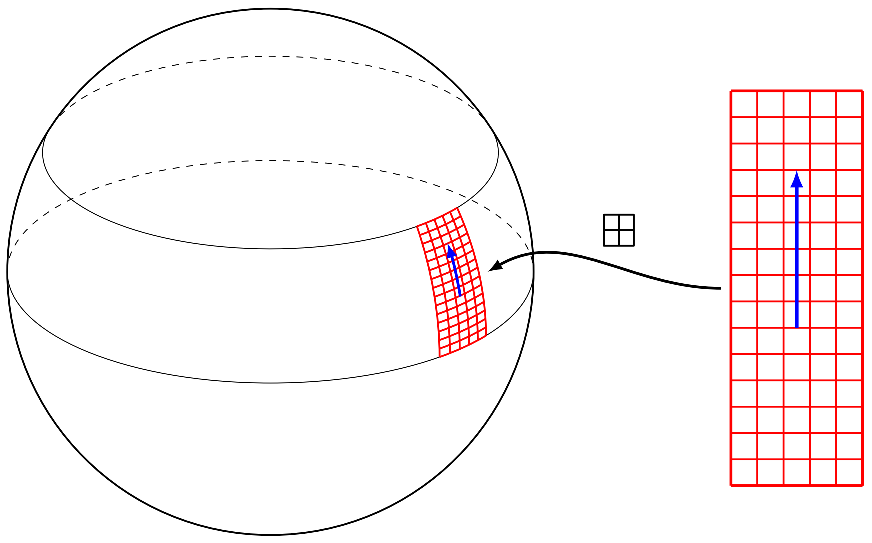


Figure 3.6.: Example of the application of the $\boxplus$-operator between a Euclidean space and a manifold. The $\boxplus$-operator maps the blue vector from the rectilinear grid to the sphere.

have to redefine some of the terms in the NLLSQ estimation. We rewrite the error function (cf. (3.1)) as

$$\boldsymbol{e}_i(\breve{\boldsymbol{x}}) = \boldsymbol{h}_i(\breve{\boldsymbol{x}}) \boxminus \boldsymbol{z}_i \tag{3.30}$$

and its Taylor expansion as

$$\boldsymbol{e}_i(\breve{\boldsymbol{x}} \boxplus \Delta\boldsymbol{x}) \approx \boldsymbol{e}_i(\breve{\boldsymbol{x}}) + \breve{\boldsymbol{J}}_i \Delta\boldsymbol{x}, \tag{3.31}$$

$$\breve{\boldsymbol{J}}_i = \left.\frac{\partial \boldsymbol{e}_i(\breve{\boldsymbol{x}} \boxplus \Delta\boldsymbol{x})}{\partial \Delta\boldsymbol{x}}\right|_{\Delta\boldsymbol{x}=0}. \tag{3.32}$$

Note that the initial guess $\breve{\boldsymbol{x}}$ spans over the overparametrized space but that the increments $\Delta\boldsymbol{x}$ (and accordingly, $\Delta\boldsymbol{x}^*$) are expressed in a minimal representation and computed in the local Euclidean neighborhood of $\breve{\boldsymbol{x}}$. Therefore, we remap them into the original space via

$$\boldsymbol{x}^* = \breve{\boldsymbol{x}} \boxplus \Delta\boldsymbol{x}^*. \tag{3.33}$$

In total, we observe that performing the optimization on a manifold consists in defining two operators $\boxplus$, $\boxminus$, and the Jacobian $\breve{\boldsymbol{J}}$ and plugging them into the equations for the

Euclidean space. Specifically, we highlight that the structure of $\boldsymbol{H}$ stays identical as the structure of $\breve{\boldsymbol{J}}$ is untouched. In this thesis we deal with problems in $SO(2)$ where the encapsulation effectively results in appropriately normalizing the rotations. The operator $\boxplus$ is simply defined as

$$\breve{\boldsymbol{x}} \boxplus \Delta\boldsymbol{x}^* = \mathrm{normalize}_{[-\pi,\pi)}(\breve{\boldsymbol{x}} + \Delta\boldsymbol{x}^*), \tag{3.34}$$

where the function $\mathrm{normalize}_{[-\pi,\pi)}(\boldsymbol{x})$ normalizes the rotational component of $\boldsymbol{x}$ to $[-\pi, \pi)$. We do not explicitly perform our derivations on manifolds in the following to leave the notation uncluttered. However, our approach is easily generalized to other manifolds such as $SO(3)$ by replacing the Euclidean operators $+, -$ with $\boxplus, \boxminus$ where appropriate.

3.4.5. Probabilistic interpretation

In this section we show that the NLLSQ estimator is a ML estimator and that the system matrix is the information matrix of the state given all measurements. Both statements require certain assumptions. To show the first statement, we assume the measurement errors distributed according to the AWGN assumption, i.e.,

$$\boldsymbol{z}_i = \boldsymbol{h}_i(\boldsymbol{x}) + \boldsymbol{\epsilon}_i, \tag{3.35}$$

where $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_i)$. With these assumptions the probability of a single measurement given $\boldsymbol{x}$ is

$$p(\boldsymbol{z}_i|\boldsymbol{x}) = \eta_i \exp\left(-\frac{1}{2}(\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}))^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}))\right). \tag{3.36}$$

If we additionally assume the measurement errors to be uncorrelated, we obtain the probability for all measurements given $\boldsymbol{x}$ as

$$p(\boldsymbol{z}|\boldsymbol{x}) = \prod_{i=1}^{n} p(\boldsymbol{z}_i|\boldsymbol{x}). \tag{3.37}$$

We seek the state $\boldsymbol{x}^*$ that maximizes $p(\boldsymbol{z}|\boldsymbol{x})$. This is equivalent to seeking the state $\boldsymbol{x}^*$ that minimizes the negative log likelihood, i.e.,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} p(\boldsymbol{z}|\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \left(-\log p(\boldsymbol{z}|\boldsymbol{x})\right). \tag{3.38}$$

Let us denote the inverse of the measurement covariance matrix as $\boldsymbol{\Sigma}_i^{-1} = \boldsymbol{\Lambda}_i$. We see that solving (3.38) is the same as minimizing the global error function as defined in the nonlinear least squares problem in (3.5) because

$$\begin{aligned}
-\log p(\boldsymbol{z}|\boldsymbol{x}) &= -\log \prod_{i=1}^{n} p(\boldsymbol{z}_i|\boldsymbol{x}) && (3.39)\\
&= -\log \prod_{i=1}^{n} \eta_i \exp\left(-\frac{1}{2}(\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}))^\top \boldsymbol{\Lambda}_i (\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}))\right) && (3.40)\\
&= -\sum_{i=1}^{n} \log\left[\eta_i \exp\left(-\frac{1}{2}(\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}))^\top \boldsymbol{\Lambda}_i (\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}))\right)\right] && (3.41)\\
&= \eta' + \sum_{i=1}^{n} \frac{1}{2}(\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x}))^\top \boldsymbol{\Lambda}_i (\boldsymbol{z}_i - \boldsymbol{h}_i(\boldsymbol{x})). && (3.42)
\end{aligned}$$

Therefore, we maximize the likelihood of the data when minimizing the global error function, i.e.,

$$\arg\max_{\boldsymbol{x}} p(\boldsymbol{z}|\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}). \tag{3.43}$$

In other words the NLLSQ optimization maximizes the joint likelihood of the measurements given certain error assumptions.

Furthermore, Kümmerle (2013) shows that under certain assumptions the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ corresponds to the Gaussian distribution $\mathcal{N}(\boldsymbol{x}^*, \boldsymbol{H}^{-1})$. In formal notation, $p(\boldsymbol{x}|\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{x}^*, \boldsymbol{H}^{-1})$. This means that the system matrix $\boldsymbol{H}$ is the information matrix for the state given all measurements. This useful property allows us to compute the covariance of each state variable by computing the respective block in $\boldsymbol{H}^{-1}$. To this end, we give in Section 5.6 an algorithm that exploits the sparse Cholesky decomposition of $\boldsymbol{H}$.

3.4.6. Graph-based representations

Throughout this thesis we make extensive use of pose graphs. Therefore, we explain in this section the main concepts behind them and the relation to some other graphical models.

Probabilistic graphical models

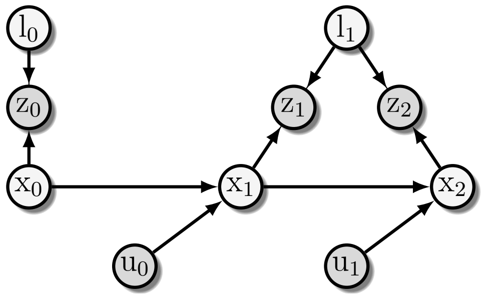
Probabilistic graphical models are probabilistic models that can be represented by graphs. In these graphs, the conditional dependencies between random variables are modeled with nodes and edges. Therefore, the graph structure represents a factorization of the joint probability distribution over all random variables. A common task within these graphs is inference, i.e., fitting the observed data to find the most likely values for its underlying quantities of interest. DBNs, factor graphs, and MRFs are particular types of probabilistic graphical models. Figure 3.7 shows a graphical model in its DBN, factor graph, and MRF representation. We refer the reader for more information on this topic to Koller and Friedman (2009).

DBNs (Dean and Kanazawa, 1988) are directed acyclic graphs where nodes represent time-dependent random variables and edges represent conditional dependencies between them. In a DBN one distinguishes between *observable* and *unobservable* (also called *hidden* or *latent*) variables. Observed nodes in Figure 3.7a are displayed as gray nodes while unobservable nodes are shown as white nodes.

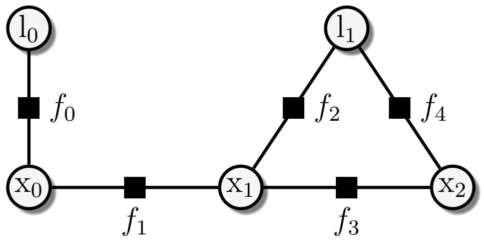
Factor graphs are bipartite undirected graphs and have been proposed by Kschischang et al. (2001). They serve as a general tool to break down functions with many variables into smaller subsets of variables. Take the product

$$f(x_0, x_1, x_2, l_0, l_1) = f_0(x_0, l_0) f_1(x_0, x_1) f_2(x_1, l_1) f_3(x_1, x_2) f_4(x_2, l_1), \tag{3.44}$$

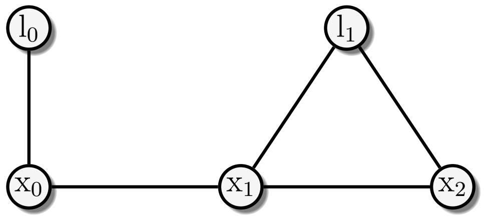
for example, which is represented by Figure 3.7b. This is called factorization and is especially useful for probabilistic problems. In these problems a factor graph decomposes a joint probability distribution into factors that depend only on a subset of the random variables. These factors represent functions over the connected variables. They are usually represented as undirected graphs with circular nodes for the variables and square nodes for the factors. The factors are connected with edges to each variable that



(a) DBN.



(b) Factor graph.



(c) MRF.

Figure 3.7.: Equivalent representations of a DBN, a factor graph, and a MRF. The toy example depicts a SLAM problem in which a robot estimates its own trajectory (nodes x_0, x_1, and x_2) and the position of two landmarks (nodes l_0 and l_1).

their underlying function relies on. Note that so-called prior factors only influence a single variable and appear therefore as open-ended factors. Overall, edges make the conditional dependencies between random variables apparent.

A MRF is described by an undirected graph. It models a set of random variables that have the Markov property; that is, a node is conditionally independent of all other nodes given its direct neighbors. The graphical structure of a MRF is typically similar to that of the corresponding factor graph except that the factors are omitted. This means that edges always connect variable nodes directly. As a consequence, prior factors cannot be modeled. Conditional random fields are a particular kind of MRFs in which each random variable may be additionally conditioned upon a set of global observations.

Pose graphs are a particular kind of factor graph in which nodes represent poses and edges represent spatial constraints between them. They have initially been proposed in the context of SLAM problems. In their seminal work Lu and Milios (1997) define an offline optimization method where they model poses and the constraints between them as a *network of pose relations*. Later works built upon this intuition and paved the way to the current understanding of the graph-based SLAM formulation (Gutmann and Konolige, 2000; Frese and Hirzinger, 2001; Konolige, 2004). Folkesson and Christensen (2004) first refer to it as *Graphical SLAM*, and Thrun and Montemerlo (2006) propose their *GraphSLAM* algorithm. As landmarks or features are part of many SLAM problems, these graphs are strictly speaking not pose graphs but rather *pose/feature graphs* as Olson (2008) calls them. They can be represented as factor graphs. The term *pose graph* (Olson et al., 2006; Eustice et al., 2006b; Folkesson and Christensen, 2007) was coined to refer to SLAM graphs in which all nodes represent poses.

Nowadays, pose/feature graphs and pose graphs are the prevailing paradigm for SLAM problems (Stachniss et al., 2016; Grisetti et al., 2010a). When building up the graph, one can directly obtain a pose graph instead of a pose/feature graph representation by matching the sensor observations to spatial constraints between nodes. Alternatively, one can marginalize all features from an existing pose/feature graph to obtain the corresponding pose graph. Figure 3.8 shows the pose graph that corresponds to the factor graph in Figure 3.7 after marginalizing all landmark observations. There exist multiple software frameworks that can aid the development of graph-based optimization algorithms, including General Graph Optimization (g2o) by Kümmerle et al. (2011), Sparse Sparse Bundle Adjustment (sSBA) by Konolige (2010), Georgia Tech Smoothing and

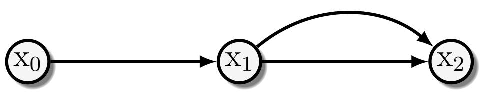


Figure 3.8.: An example pose graph. It represents the same problem as in Figure 3.7 but the landmark observations have been converted to edges between nodes by marginalization.

Mapping library (GTSAM) by Dellaert (2012), and Incremental Smoothing and Mapping (iSAM) by Kaess et al. (2008, 2012).

In this thesis we base our methods on pose graphs as they are a well-suited tool for modeling estimation problems that only contain poses and relations between them. We do not need the expressiveness of factor graphs or the probabilistic descriptiveness of MRFs. Pose graphs offer insight over the relation of state variables and their constraints, and are a well-understood representation of state estimation problems.

Pose graphs

Formally, a pose graph is defined as a directed graph $\overrightarrow{\mathcal{G}} = (\mathcal{X}, \overrightarrow{\mathcal{Z}})$ where $\mathcal{X} = \{\mathrm{x}_i\}_{i=1}^{m}$ is the set of nodes and $\overrightarrow{\mathcal{Z}} = \{\mathrm{z}_i\}_{i=1}^{n}$ is the set of directed edges. For the purpose of this thesis we consider all pose graph edges to be binary[2]. The node x_i represents the i-th pose. An edge z_i represents a spatial constraint between the connected nodes. It embodies a probability distribution over the relative transformations between the connected nodes. This probability distribution is assumed to be of Gaussian nature. It is therefore completely defined by the mean estimate and covariance matrix. In graphical representations we will usually omit the explicit indication of the mean and covariance.

Looking at the term “constraint” from a perspective of the field of constrained optimization, we note that it refers to a *soft constraint* (Olson, 2008): the objective function is penalized to the extent that these soft constraints are violated, but the optimization solution is still valid. This is in contrast to *hard constraints* that are required to be satisfied for the solution to be valid. The edge z_i represents the i-th soft constraint. With a slight abuse of notation we note both the i-th edge and constraint as z_i. The intuition behind modeling a soft constraint z_i as *directed* edge is that it describes the movement that one

[2]The sole exception to this is a discussion in Section 6.4.2 about how constraints with autocorrelated noise can be modeled.

has to perform to go from x_i to x_{i+1}. Therefore, it is a natural representation of spatial constraints. Other edges, such as loop closure edges, build up on the same concept: they describe the relative transformation that a starting node would have to undergo to transform into the target node.[3]

Typically, these edges arise from measurements. An individual measurement ideally leads to the creation of a single edge in the graph. We will see in Section 5.2 that this assumption is commonly violated in real systems as measurements generally are not made at exactly the same time steps. Edges can also arise from other kind of information. A canonical example are loop closure edges that arise whenever a robot recognizes that he revisits the same location. The robot has made a virtual measurement to relate the two locations to each other. Moreover, edges can also represent prior information on the structure of nodes. A robot could recognize and move alongside walls and add the constraint that these man-made structures are usually perpendicular to each other, for example. Therefore, it is important to make the distinction between measurements and edges/soft constraints.

There are two types of nodes: one represent observed poses at certain timestamps and the others are state variables whose values we wish to determine. We call the former *observed nodes* and denote them with $\mathrm{x}_i^{\mathrm{w}}$. In the spirit of DBNs we refer to the latter as *hidden nodes*. The state variables $\boldsymbol{x}_i$ that correspond to these hidden nodes are the ones whose value we are interested in. They are displayed as circles with a black border. In contrast, observed nodes are displayed with a colored border and denoted as $\mathrm{x}_i^{\mathrm{w}}$. They represent the same kind of soft constraint that prior factors represent in factor graphs, i.e., constraints that act only upon a single hidden node. An alternative but less thorough way of displaying them would be as open-ended edges (similar to unary factors in factor graphs). They are *fixed* in the optimization problem so that their value is not changed during the optimization, but they still constrain the connected hidden nodes in the global coordinate frame. We detail this behavior after relating pose graphs to the underlying optimization problem. Figure 3.9 shows a pose graph with the hidden nodes x_0 to x_5 and the observed node $\mathrm{x}_0^{\mathrm{w}}$.

Let us examine the relation of pose graphs and NLLSQ problems. Usually, we are interested in inference over the pose graph to find the values of the state variables that

[3]The direction of the edge is important as the covariance matrix of the corresponding soft constraint is accordingly rotated, see the definition of the error functions.

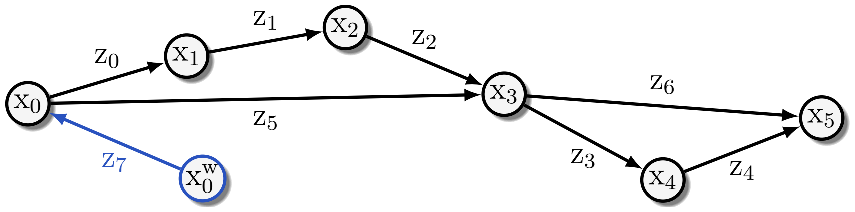


Figure 3.9.: An example pose graph with a single observed node $\mathrm{x}_0^{\mathrm{w}}$. Each edge z_0 to z_7 represents a soft constraint. We are interested in the poses of x_0 to x_5.

best fit all constraints, where the constraints arise from a set of measurements. Each constraint z_i imposes a relation between the state of the model (that is, the poses of the hidden nodes) and some observed quantities:

$$\boldsymbol{h}_i(\boldsymbol{x}) = \boldsymbol{z}_i. \tag{3.45}$$

Here, we pick up the notation from Section 3.4.1. As stated above, in a pose graph we treat constraints that either relate two hidden nodes or that effect only a single hidden node. We call the former *odometry constraints* and the latter *global pose constraints*. Accordingly, the function $\boldsymbol{h}_i$ depends either on two hidden nodes or on a single hidden node. For notational simplicity we use the entire state vector $\boldsymbol{x}$ as argument. Deviations from these constraints result in error terms, see (3.1). The initial configuration of the nodes corresponds to the initial guess $\breve{\boldsymbol{x}}$ in the Gauss-Newton optimization. This allows us to construct the global error function $F(\boldsymbol{x})$ (cf. (3.4)) for a given pose graph. By making certain assumptions about the measurements and their errors we have shown in Section 3.4.5 that the ML estimate is given by solving the corresponding NLLSQ problem. Therefore, we can infer the most likely configuration of the nodes in a pose graph by solving the corresponding NLLSQ problem. Phrased differently, if we have a NLLSQ problem at hand in which the state variables $\boldsymbol{x}_i$ represent poses, then we can represent it as a pose graph. The graph-based representation allows us to visualize and better understand the dependencies of the state variables.

Constraints between hidden nodes can for instance be constructed from odometry information. For a constraint z_k from hidden node x_i to x_j, the function $\boldsymbol{h}_k^{\mathrm{v}}(\boldsymbol{x})$ expresses the predicted pose. It is defined by

$$\boldsymbol{h}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{bmatrix} \boldsymbol{R}_{\theta_i}^{\top} \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right) \\ \theta_j - \theta_i \end{bmatrix} \tag{3.46}$$

where $\boldsymbol{R}_\phi$ is the standard two-dimensional rotation matrix

$$\boldsymbol{R}_\phi = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{3.47}$$

The corresponding error function $\boldsymbol{e}_k(\boldsymbol{x}, \boldsymbol{z}_k^{\mathrm{v}}) = \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})$ is the common error function (Grisetti et al., 2010a; Kümmerle et al., 2011) between two poses with

$$\boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}}) = \begin{bmatrix} \boldsymbol{R}_{\Delta\theta_k^{\mathrm{v}}}^{\top} \left(\boldsymbol{R}_{\theta_i}^{\top} \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right) - \begin{bmatrix} \Delta x_k^{\mathrm{v}} \\ \Delta y_k^{\mathrm{v}} \end{bmatrix} \right) \\ \theta_j - \theta_i - \Delta\theta_k^{\mathrm{v}} \end{bmatrix}. \tag{3.48}$$

This is conform with our definition in (3.1) except that the additional application of $\boldsymbol{R}_{\Delta\theta_k^{\mathrm{v}}}^{\top}$ serves to rotate the error vector into the frame of the predicted pose. This is necessary because the information matrix $\boldsymbol{\Lambda}_k^{\mathrm{v}}$ is given in that frame. Both have to be expressed in the same frame such that the weighted squared error terms (cf. (3.3)) are correctly scaled[4]. For the construction of the NLLSQ problem we also need the corresponding partial derivatives of this error function

$$\frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_i} = \begin{bmatrix} -\boldsymbol{R}_{\Delta\theta_k^{\mathrm{v}}}^{\top} \boldsymbol{R}_{\theta_i}^{\top} & \boldsymbol{R}_{\Delta\theta_k^{\mathrm{v}}}^{\top} \frac{\partial \boldsymbol{R}_{\theta_i}^{\top}}{\partial \theta_i} \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right) \\ \mathbf{0}^{\top} & -1 \end{bmatrix}, \tag{3.49}$$

$$\frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_j} = \begin{bmatrix} \boldsymbol{R}_{\Delta\theta_k^{\mathrm{v}}}^{\top} \boldsymbol{R}_{\theta_i}^{\top} & 0 \\ \mathbf{0}^{\top} & 1 \end{bmatrix}, \tag{3.50}$$

[4] Alternatively, it is equivalent to define $\boldsymbol{\Lambda}_k^{\mathrm{v}}$ in the frame of x_i, or to apply the rotations to $\boldsymbol{\Lambda}_k^{\mathrm{v}}$ directly, such that $\tilde{\boldsymbol{\Lambda}}_k^{\mathrm{v}} = \boldsymbol{R}_{\Delta\theta_k^{\mathrm{v}}} \boldsymbol{\Lambda}_k^{\mathrm{v}} \boldsymbol{R}_{\Delta\theta_k^{\mathrm{v}}}^{\top}$. Then the relation $\boldsymbol{h}_i(\boldsymbol{x}) = \boldsymbol{z}_i$ is met more strictly. However, we prefer to stay consistent with the prevalent notation in graph-based SLAM at the expense of this slight abuse in the definition of $\boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})$.

where $\mathbf{0}^\top = \begin{bmatrix}0, 0\end{bmatrix}$.

Constraints from observed to hidden nodes behave differently in that the observed nodes are not part of the optimization problem. Instead, their pose is known and they are thus fixed. The function $\boldsymbol{h}_k^{\mathrm{w}}(\boldsymbol{x}_i)$, that predicts the pose of node x_i, is simply equal to this pose:

$$\boldsymbol{h}_k^{\mathrm{w}}(\boldsymbol{x}_i) = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}. \tag{3.51}$$

Given the global pose estimate $\boldsymbol{z}_k^{\mathrm{w}}$ of the constraint z_k, the error function for constraints originating from observed nodes $\boldsymbol{e}_k(\boldsymbol{x}, \boldsymbol{z}_k^{\mathrm{w}}) = \boldsymbol{e}_k^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}})$ is given by

$$\boldsymbol{e}_k^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}}) = \begin{bmatrix} \boldsymbol{R}_{\theta_k^{\mathrm{w}}}^\top & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} (\boldsymbol{x}_i - \boldsymbol{z}_k^{\mathrm{w}}). \tag{3.52}$$

Again, the application of $\boldsymbol{R}_{\theta_k^{\mathrm{w}}}^\top$ serves to rotate the error vector into the same frame as the corresponding information matrix. In contrast to $\boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})$, for $\boldsymbol{e}_i^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}})$ we are only interested in its partial derivative with respect to the connected hidden node because the observed node is not being estimated in the optimization problem. This derivative is

$$\frac{\partial \boldsymbol{e}_k^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}})}{\partial \boldsymbol{x}_i} = \begin{bmatrix} \boldsymbol{R}_{\theta_k^{\mathrm{w}}}^\top & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{3.53}$$

With these definitions at hand, we are able to construct the NLLSQ problem for a given pose graph[5].

Intuitively, the graph-based representation of the NLLSQ problem can be understood as a mass-spring model (Golfarelli et al., 1998; Barfoot, 2017), where the nodes represent masses and the edges represent springs. The optimal posterior solution of the graph corresponds to the minimum energy state of the system with respect to the en-

[5]It is equivalent to derive $\boldsymbol{e}_k^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}})$ as a special case of $\boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_k, \boldsymbol{x}_i, \begin{bmatrix}0,0,0\end{bmatrix}^\top)$. For this we temporarily consider both x_i and x_k as hidden nodes, initialize the pose of x_k with $\boldsymbol{z}_k^{\mathrm{w}}$, and create an edge z_j with a spatial transformation of $\begin{bmatrix}0,0,0\end{bmatrix}^\top$ from x_k to x_i. Then, we compute $\boldsymbol{H}$ and $\boldsymbol{b}$, and subsequently condition them on $\boldsymbol{x}_k$. This conditioning effectively fixes x_k and suppresses the k-th block row and column from $\boldsymbol{H}$ and $\boldsymbol{b}$ as they are given in information form. While this derivation is less intuitive, it is useful for the implementation as we can use the same error functions for constraints from observed nodes and between hidden nodes.

ergy stored in the springs. The observed nodes “pull” the hidden nodes towards them because their constraints equal zero if and only if the pose of the hidden nodes are identical to the pose of the connected observed nodes. Similarly, the edges between hidden nodes “push” or “pull” the hidden nodes (depending on their current configuration) to relative poses which are equal to the relative transformation encoded in the edge. The optimization of the graph therefore seeks its state of maximum relaxation.

The matrix $\boldsymbol{H}$ is a symmetric block matrix. All blocks have the same size, which is in our case 3×3. This is due to parametrizing a single state as $(x, y, \theta)^\top$. The block structure of $\boldsymbol{H}$ in NLLSQ optimization is equal to the adjacency matrix of its corresponding pose graph (Kümmerle, 2013). This is an important property because it means that we influence the block structure of $\boldsymbol{H}$ by designing the corresponding graph structure. This is interesting because the block structure of $\boldsymbol{H}$ dominates the computational complexity of the optimization problem. We will show in Section 5.2 how to exploit this knowledge to construct a structure that is beneficial for our problem.

To understand how an edge leads to an entry in $\boldsymbol{H}$, let us first consider the general case of an edge z_k that connects x_i to x_j. In (3.18) we derived that $\boldsymbol{H}$ depends entirely on the Jacobian and information matrix of the constraints. In turn, the Jacobian of the constraint contains only nonzero elements in block columns i and j:

$$\breve{\boldsymbol{J}}_k^{\mathrm{v}} = \left[\begin{array}{ccccccccccc} & & & i & & & & j & & & \\ \mathbf{0} & \cdots & \mathbf{0} & \left.\frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} & \mathbf{0} & \cdots & \mathbf{0} & \left.\frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_j}\right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} & \mathbf{0} & \cdots & \mathbf{0} \end{array}\right]. \tag{3.54}$$

Thus, the edge between x_i and x_j leads to additional block entries in $\boldsymbol{H}_{ii}, \boldsymbol{H}_{ij}, \boldsymbol{H}_{ji}$, and $\boldsymbol{H}_{jj}$. With a similar argument, we see that it leads to entries in the i-th and j-th block row of $\boldsymbol{b}$. Let $\boldsymbol{H}_k = (\breve{\boldsymbol{J}}_k^{\mathrm{v}})^\top \boldsymbol{\Lambda}_k^{\mathrm{v}} \breve{\boldsymbol{J}}_k^{\mathrm{v}}$ and $\boldsymbol{b}_k = (\breve{\boldsymbol{J}}_k^{\mathrm{v}})^\top \boldsymbol{\Lambda}_k^{\mathrm{v}} \breve{\boldsymbol{e}}_k^{\mathrm{v}}$. They are given by

$$\boldsymbol{H}_k = \begin{bmatrix} \ddots & & & & \\ & \boldsymbol{H}_{ii} & \cdots & \boldsymbol{H}_{ij} & \\ & \vdots & \ddots & \vdots & \\ & \boldsymbol{H}_{ji} & \cdots & \boldsymbol{H}_{jj} & \\ & & & & \ddots \end{bmatrix}, \tag{3.55}$$

with the matrix entries

$$\boldsymbol{H}_{ii} = \left(\left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_i} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} \right)^{\top} \boldsymbol{\Lambda}_k^{\mathrm{v}} \left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_i} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}}, \tag{3.56}$$

$$\boldsymbol{H}_{ij} = \left(\left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_i} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} \right)^{\top} \boldsymbol{\Lambda}_k^{\mathrm{v}} \left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_j} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}}, \tag{3.57}$$

$$\boldsymbol{H}_{ji} = \left(\left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_j} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} \right)^{\top} \boldsymbol{\Lambda}_k^{\mathrm{v}} \left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_i} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}}, \tag{3.58}$$

$$\boldsymbol{H}_{jj} = \left(\left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_j} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} \right)^{\top} \boldsymbol{\Lambda}_k^{\mathrm{v}} \left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_j} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}}. \tag{3.59}$$

The right-hand side vector is defined by

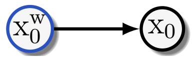
$$\boldsymbol{b}_k = \begin{bmatrix} \vdots \\ \left(\left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_i} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} \right)^{\top} \boldsymbol{\Lambda}_k^{\mathrm{v}} \breve{\boldsymbol{e}}_k^{\mathrm{v}} \\ \vdots \\ \left(\left. \frac{\partial \boldsymbol{e}_k^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{v}})}{\partial \boldsymbol{x}_j} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} \right)^{\top} \boldsymbol{\Lambda}_k^{\mathrm{v}} \breve{\boldsymbol{e}}_k^{\mathrm{v}} \\ \vdots \end{bmatrix}. \tag{3.60}$$

For constraints between successive nodes (that is, $j = i + 1$) this results in 2×2 block entries in $\boldsymbol{H}$. In contrast, for a constraint z_k from the observed node $\mathrm{x}_k^{\mathrm{w}}$ to the hidden node x_i the Jacobian contains only a single partial derivative as $\mathrm{x}_k^{\mathrm{w}}$ is not part of the state vector $\boldsymbol{x}$:

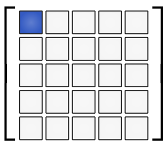
$$\breve{\boldsymbol{J}}_k^{\mathrm{w}} = \begin{array}{c} \begin{matrix} & & & i & & & \end{matrix} \\ \begin{bmatrix} 0 & \cdots & 0 & \left. \frac{\partial \boldsymbol{e}_k^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}})}{\partial \boldsymbol{x}_i} \right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} & 0 & \cdots & 0 \end{bmatrix} \end{array}. \tag{3.61}$$

As a result, this only leads to block entries in $\boldsymbol{H}_{ii}$ and $\boldsymbol{b}_i$. Figure 3.10 illustrates the construction of a pose graph and the resulting structure of $\boldsymbol{H}$.

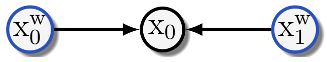
We now know how nodes and edges lead to entries in $\boldsymbol{H}$ and $\boldsymbol{b}$. Looking at this from a different perspective we can now infer how certain operations on $\boldsymbol{H}$ and $\boldsymbol{b}$ can be represented by nodes and edges. For this we analyze the structure and the content of $\boldsymbol{H}$ and $\boldsymbol{b}$ after applying these operations and relate the changes to those caused by nodes and edges. That is, we will see in Section 5.3 how marginalization can be understood as adding and removing certain nodes and edges. Moreover, we will derive in Section 6.4



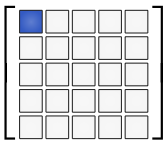
(a) We are interested in the pose $\boldsymbol{x}_0$ that corresponds to the hidden node x_0. Optimizing the graph leads to x_0 being identical to the pose of x_0^w, as there are no other constraints.



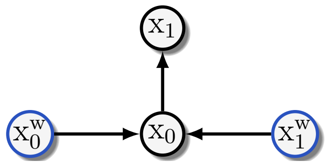
(b) Sparsity pattern of the system matrix $\boldsymbol{H}$ for the graph in (a). The observed node x_0^w influences the upper left block matrix of $\boldsymbol{H}$ that solely conveys information about $\boldsymbol{x}_0$.



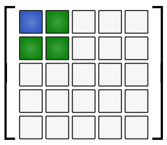
(c) Adding the observed node x_1^w results in conflicting information for x_0.



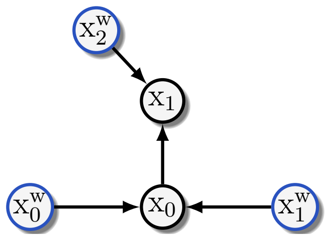
(d) The value of the upper left block matrix $\boldsymbol{H}$ changes, but the structure of $\boldsymbol{H}$ rests unaffected.



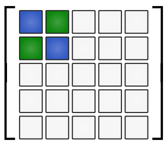
(e) We add the information that x_1 is a certain distance away from x_0.



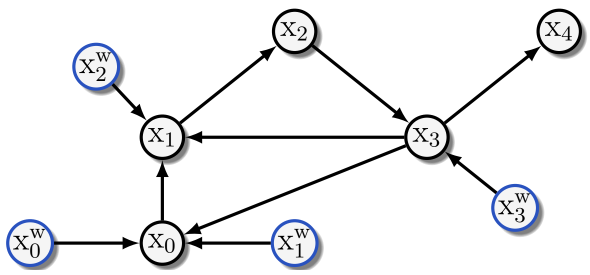
(f) The new constraint leads to diagonal and off-diagonal block entries in $\boldsymbol{H}$.



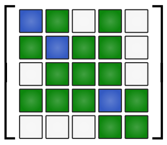
(g) The additional observed node x_2^w constraints x_1 globally. By that, it influences the pose of x_0 after the optimization, too.



(h) Adding x_2^w changes a block entry in $\boldsymbol{H}$.



(i) Finally, additional hidden and observed nodes were added and the graph is completely constructed.



(j) Sparsity pattern of the system matrix $\boldsymbol{H}$ for the graph in (i).

Figure 3.10.: Illustration of a pose graph construction for a single time step and the influence on the system matrix $\boldsymbol{H}$.

how autocorrelated errors can be modeled with graph elements. This makes it necessary to treat the pose graph as a *hypergraph*, in which an (hyper)edge can join any number of nodes. However, for the overwhelming majority of this thesis, we refer to pose graphs with directed binary edges.

3.4.7. Relation to geodetic mapping

Historically, least squares estimation was developed for geodetic and astronomic purposes by Legendre and Gauss between 1795 and 1823 (Legendre, 1805; Gauss, 1823). Gauss applied it successfully to both fieldsAgarwal et al. (2014b) by predicting the location of the asteroid Ceres from observations and estimating the geometric relations in triangulation networks during the famous triangulation of the Kingdom of Hanover between 1821 and 1825 (Hald, 2008; Stigler, 2007). In the middle of the 20th century the photogrammetry community adopted essentially the same estimation methodology in the form of bundle adjustment (Triggs et al., 2000).

Despite tackling similar problems it seems that the mobile robotics community developed early solutions to the SLAM problem largely independently of classical adjustment theory, starting from a probabilistic view (Thrun et al., 2005). With the advent of graph-based approaches optimization-based methods are currently established as the de facto standard for SLAM problems. They show close resemblance to adjustment models in their mathematical formulations. Indeed, Förstner (2013) shows that graphical models (i.e., Bayes nets, MRFs, and factor graphs) are a generalization of geodetic networks and bundle adjustments. The solution of adjustment models can be interpreted as inference over the corresponding probabilistic models. This leads to the classical (nonlinear) least squares problem of adjustment theory. The same holds true for the solution of pose graphs, where the NLLSQ problem arises as solution to the inference problem over factor graphs. Therefore, we deem it important to highlight the relation between the mathematical formulations of the two NLLSQ problems.

The most obvious barrier of translating between graph-based optimization and adjustment theory is presumably the different point of view and the resulting different notation. Thus, we briefly present the equivalent description and notation of the NLLSQ estimation from the point of view of geodetic mapping.

The graph-based representation in Section 3.4.6 can be linked to the well-known normal equation matrices of adjustment models (Niemeier, 2008; Förstner and Wrobel, 2016). Here, $\boldsymbol{X}$ is the parameter vector, and $\hat{\boldsymbol{X}}$ represents the adjusted parameter vector that we want to infer from the vector of observations $\boldsymbol{L}$. It has the associated covariance matrix $\boldsymbol{\Sigma}_{ll}$ which constitutes the stochastic model together with the cofactor σ_0^2 and the cofactor matrix $\boldsymbol{Q}_{ll}$, such that

$$\boldsymbol{\Sigma}_{ll} = \sigma_0^2 \boldsymbol{Q}_{ll}. \tag{3.62}$$

The inverse of the cofactor matrix is the information matrix

$$\boldsymbol{Q}_{ll}^{-1} = \boldsymbol{\Lambda}. \tag{3.63}$$

In adjustment models this matrix is typically called weighting matrix $\boldsymbol{P}$. The shortened measurement vector

$$\boldsymbol{l} = \boldsymbol{L} - \boldsymbol{L}_0 \tag{3.64}$$

is the difference between the observations $\boldsymbol{L}$ and the approximated observation vector $\boldsymbol{L}_0$. The adjusted observation vector $\hat{\boldsymbol{L}}$ is equal to the sum of the vector of observations and the vector of improvements $\boldsymbol{v}$, i.e.,

$$\hat{\boldsymbol{L}} = \boldsymbol{L} + \boldsymbol{v}. \tag{3.65}$$

The functional model relates the parameters to the observations:

$$\boldsymbol{L} + \boldsymbol{v} = f(\hat{\boldsymbol{X}}). \tag{3.66}$$

Linearizing $f(\hat{\boldsymbol{X}})$ by a first-order Taylor approximation around $\boldsymbol{X}_0$ leads to

$$f(\boldsymbol{X}_0 + \boldsymbol{x}) \approx f(\boldsymbol{X}_0) + \left.\frac{\partial f}{\partial \boldsymbol{X}}\right|_{\boldsymbol{X}=\boldsymbol{X}_0} \boldsymbol{x}. \tag{3.67}$$

The partial derivatives $\frac{\partial f}{\partial \boldsymbol{X}}\big|_{\boldsymbol{X}=\boldsymbol{X}_0}$ are equivalent to the Jacobians $\breve{\boldsymbol{J}}_i$ (see (3.9)), and we stacked them into the design matrix $\boldsymbol{A}$. Plugging (3.67) into (3.66) leads to the matrix representation of the linearized form of the functional model:

$$\boldsymbol{l} + \boldsymbol{v} = \boldsymbol{A}\hat{\boldsymbol{x}}. \tag{3.68}$$

The least squares adjustment requires iteratively choosing an initial parameter vector $\boldsymbol{X}_0$, linearizing the functional model, and estimating the parameter corrections $\hat{\boldsymbol{x}}$ by solving

$$\boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A} \hat{\boldsymbol{x}} = \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{l}. \tag{3.69}$$

Note that we can identify $\boldsymbol{H}$ with $\boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A}$ and $\boldsymbol{b}$ with $-\boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{l}$ by comparing (3.23) with (3.69). Indeed, both approaches solve the same problem, but notation, wording, and formulations are different. For further information on the relationship of graph-based optimization in the robotics community to geodetic mapping approaches, we refer the reader to Agarwal et al. (2014a,b). In the remainder of this thesis, we stick with the graph-based notion on the NLLSQ method as it provides a way of visually reasoning about the relations of observations to parameters.

3.5. Covariance Intersection

CI is an algorithm to fuse two state estimates with cross-correlated but unknown noise. This is helpful for generic pose fusion where the cross-correlation is typically unknown. In the following we present in Section 3.5.1 the optimal and naive fusion, followed in Section 3.5.2 by the description of the general CI approach. In Section 3.5.3 and Section 3.5.4 we detail how to derive the closed-form solutions for it. This derivation is extended in Section 6.3 where we use CI to deal with cross-correlated errors.

3.5.1. Optimal and naive fusion of states with cross-correlated noise

In state estimation and fusion problems it is a common task to fuse two state estimates into a single state. This is conventionally achieved by a linear combination of the two input state estimates. If the correlation between the two input state estimates is known, we can derive the optimal fusion result. For this we follow the derivation of Li et al. (2015); Chen et al. (2002).

Two normally distributed states $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ with covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ can generally be combined by

$$\boldsymbol{x} = \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{A}_2 \boldsymbol{x}_2 \tag{3.70}$$

where $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ denote quadratic matrices. To ensure that the mean error is equal to zero, $\boldsymbol{A}_1 + \boldsymbol{A}_2 = \boldsymbol{I}$ must hold. The variance law of error propagation implies for the resulting covariance matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^\top & \boldsymbol{\Sigma}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_1^\top \\ \boldsymbol{A}_2^\top \end{bmatrix} \tag{3.71}$$

$$= \boldsymbol{A}_1\boldsymbol{\Sigma}_1\boldsymbol{A}_1^\top + \boldsymbol{A}_1\boldsymbol{\Sigma}_{12}\boldsymbol{A}_2^\top + \boldsymbol{A}_2\boldsymbol{\Sigma}_{12}^\top\boldsymbol{A}_1^\top + \boldsymbol{A}_2\boldsymbol{\Sigma}_2\boldsymbol{A}_2^\top . \tag{3.72}$$

To obtain a fusion with a small covariance matrix, $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are set such that $\boldsymbol{\Sigma}$ is optimal in some sense, e.g., that $\boldsymbol{\Sigma}$ has a minimal trace or determinant. The optimal solution with respect to trace minimization yields

$$\boldsymbol{\Sigma} = \left(\begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^\top & \boldsymbol{\Sigma}_2 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{I} \end{bmatrix} \right)^{-1} . \tag{3.73}$$

This is the covariance of the *optimal fusion*. However, it requires knowledge about the cross-correlation matrix $\boldsymbol{\Sigma}_{12}$. If it is unknown, the *naive approach* is to assume $\boldsymbol{\Sigma}_{12} = \boldsymbol{0}$. In that case (3.73) can be simplified to

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}, \right)^{-1} \tag{3.74}$$

which corresponds to the *Kalman Gain* as defined in a classical Kalman filter. This, however, leads to overconfident estimates if $\boldsymbol{\Sigma}_{12}$ is positive definite. We use this knowledge about the optimal and naive fusion to evaluate our CI framework.

The error of the naive fusion can be estimated from (3.72) with $\left\| \boldsymbol{A}_1\boldsymbol{\Sigma}_{12}\boldsymbol{A}_2^\top + \boldsymbol{A}_2\boldsymbol{\Sigma}_{12}^\top\boldsymbol{A}_1^\top \right\|$. If the noise of the estimates correlate with a correlation coefficient ρ, i.e., $\boldsymbol{\Sigma}_{12} = \rho\boldsymbol{I}$, the error is linear in ρ. If we can find an upper bound for the cross-correlation, we can decide whether the error is small and tolerable or if we need to take the cross-correlation into account. In general, however, we need a method to produce conservative uncertainty estimates without knowledge of their cross-correlation. For this we detail the CI framework in Section 3.5.2.

3.5.2. Fusion with unknown cross-correlation

CI is the statistically optimal algorithm to fuse two estimates $\boldsymbol{x}_1, \boldsymbol{x}_2$ with associated covariance matrices $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ if the cross-correlations $\boldsymbol{\Sigma}_{12}$ between their errors are unknown (Uhlmann, 1995). The resulting covariance matrix $\boldsymbol{\Sigma}^\omega$ and the fused state $\boldsymbol{x}^\omega$ are computed according to

$$\boldsymbol{\Sigma}^\omega = \left(\omega \boldsymbol{\Sigma}_1^{-1} + (1-\omega)\boldsymbol{\Sigma}_2^{-1}\right)^{-1}, \tag{3.75}$$

$$\boldsymbol{x}^\omega = \boldsymbol{\Sigma}^\omega \left(\omega \boldsymbol{\Sigma}_1^{-1}\boldsymbol{x}_1 + (1-\omega)\boldsymbol{\Sigma}_2^{-1}\boldsymbol{x}_2\right) \tag{3.76}$$

with $\omega \in [0,1]$. The parameter ω is chosen in such a way that the covariance matrix $\boldsymbol{\Sigma}^\omega$ is minimized regarding a given objective function J such that

$$\omega^* = \underset{\omega\in[0,1]}{\arg\min}\, J(\boldsymbol{\Sigma}^\omega) \tag{3.77}$$

in order to minimize the upper bound of the corresponding mean square error matrix. Common choices for J are the trace and determinant of $\boldsymbol{\Sigma}^\omega$. We evaluate these two choices on simulated data in Section 7.4.3 and show that both choices for J are suitable for the problem of pose fusion. For low-dimensional fusion problems, Reinhardt et al. (2012) have derived closed-form solutions for both choices of J. We briefly highlight the joint diagonalization approach in Section 3.5.3 and the derivation of the closed-form solutions in Section 3.5.4.

3.5.3. Joint diagonalization

Joint diagonalization for matrices is a useful tool to derive closed-form solutions for the CI optimization. The covariance matrices are transformed in such a way that one of them becomes the identity while the other one becomes a diagonal matrix.

Let $\boldsymbol{E}_1, \boldsymbol{E}_2$ be the diagonal matrix containing the eigenvalues of $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$. The corresponding eigenvector matrices are $\boldsymbol{V}_1, \boldsymbol{V}_2$. Therefore, for $i = 1, 2$ we have

$$\boldsymbol{\Sigma}_i = \boldsymbol{V}_i \boldsymbol{E}_i \boldsymbol{V}_i^\top = \boldsymbol{V}_i \sqrt{\boldsymbol{E}_i}\sqrt{\boldsymbol{E}_i}\boldsymbol{V}_i^\top. \tag{3.78}$$

Then $\boldsymbol{\Sigma}_1$ can be transformed to the identity matrix $\boldsymbol{I}$ using $\boldsymbol{T}_1 = (\boldsymbol{V}_1\sqrt{\boldsymbol{E}_1})^{-1}$:

$$\boldsymbol{\Sigma}_1' = \boldsymbol{T}_1\boldsymbol{\Sigma}_1\boldsymbol{T}_1^\top = \boldsymbol{I}, \quad \boldsymbol{\Sigma}_2' = \boldsymbol{T}_1\boldsymbol{\Sigma}_2\boldsymbol{T}_1^\top. \tag{3.79}$$

The eigenvalue decomposition $\boldsymbol{\Sigma}_2' = \boldsymbol{V}_2'\boldsymbol{E}_2'(\boldsymbol{V}_2')^\top$ can now be used to diagonalize the second covariance matrix. With $\boldsymbol{T} = (\boldsymbol{V}_2')^\top\boldsymbol{T}_1$ we obtain

$$\boldsymbol{\Sigma}_1'' = \boldsymbol{T}\boldsymbol{\Sigma}_1\boldsymbol{T}^\top = (\boldsymbol{V}_2')^\top\boldsymbol{\Sigma}_1'\boldsymbol{V}_2' = (\boldsymbol{V}_2')^\top\boldsymbol{V}_2' = \boldsymbol{I} \tag{3.80}$$

$$\boldsymbol{\Sigma}_2'' = \boldsymbol{T}\boldsymbol{\Sigma}_2\boldsymbol{T}^\top = (\boldsymbol{V}_2')^\top\boldsymbol{T}_1\boldsymbol{\Sigma}_2(\boldsymbol{T}_1)^\top\boldsymbol{V}_2' = (\boldsymbol{V}_2')^\top\boldsymbol{\Sigma}_2'\boldsymbol{V}_2' = \boldsymbol{E}_2'. \tag{3.81}$$

We will use this result in Section 3.5.4 to derive the closed-form solutions for low-dimensional CI problems.

3.5.4. Closed-form solutions

We follow the work of Reinhardt et al. (2012) to derive the closed-form solutions. Applying the joint diagonalization on $\boldsymbol{\Sigma}^\omega$ yields

$$\boldsymbol{T}\boldsymbol{\Sigma}^\omega\boldsymbol{T}^\top = (\omega\boldsymbol{I} + (1-\omega)(\boldsymbol{E}_2')^{-1})^{-1}. \tag{3.82}$$

For $J(\cdot) = \det(\cdot)$ the minimization problem (3.77) is equivalent to

$$\omega^* = \underset{\omega\in[0,1]}{\arg\min} \det(\boldsymbol{T}\boldsymbol{\Sigma}^\omega\boldsymbol{T}^\top), \tag{3.83}$$

which follows directly from the determinant rules and that $\boldsymbol{T}$ is regular and does not depend on ω.

Let d_i be the diagonal elements of $\boldsymbol{E}_2'$, i.e., the eigenvalues of $\boldsymbol{\Sigma}_2'$ and $\bar{d}_i = \frac{1}{d_i}$. Then the minimization problem (3.83) can be expressed as

$$\omega^* = \underset{\omega\in[0,1]}{\arg\max} \prod_{i\in\mathcal{I}} (\omega + (1-\omega)\bar{d}_i) \tag{3.84}$$

where $\mathcal{I} = \{1, ..., n\}$. Using $\tilde{d}_i = \frac{\bar{d}_i}{1-\bar{d}_i}$ the explicit solution for $n = 2$ is given as

$$\omega = -\frac{1}{2}(\tilde{d}_1 + \tilde{d}_2). \tag{3.85}$$

The two candidates for $n = 3$ are

$$\omega_{1/2} = -\frac{1}{3}(\tilde{d}_1 + \tilde{d}_2 + \tilde{d}_3) \pm \sqrt{\tilde{d}_1^2 + \tilde{d}_2^2 + \tilde{d}_3^2 - \tilde{d}_1\tilde{d}_2 - \tilde{d}_1\tilde{d}_3 - \tilde{d}_2\tilde{d}_3}. \tag{3.86}$$

Note that the minimization problem (3.77) is convex on the interval $[0, 1]$ and therefore there will only be one valid solution in this interval.

Respectively, we get

$$\omega^* = \arg\min_{\omega} \sum_{i \in \mathcal{I}} \frac{a_i}{\omega + (1 - \omega)\tilde{d}_i} \tag{3.87}$$

in case of trace minimization where $a_i > 0$ denotes the i-th diagonal element of $(\boldsymbol{T}^\top)^{-1}\boldsymbol{T}^{-1}$. The roots of the derivative of the polynomial are given as

$$\omega_1 = -p + \sqrt{p^2 - q}, \text{ and } \omega_2 = -p - \sqrt{p^2 - q} \tag{3.88}$$

with

$$p = \frac{a_1\tilde{d}_2(1 + \tilde{d}_1) + a_2\tilde{d}_1(1 + \tilde{d}_2)}{a_1(1 + \tilde{d}_1) + a_2(1 + \tilde{d}_2)}, \tag{3.89}$$

$$q = \frac{a_1(\tilde{d}_2)^2(1 + \tilde{d}_1) + a_2(\tilde{d}_1)^2(1 + \tilde{d}_2)}{a_1(1 + \tilde{d}_1) + a_2(1 + \tilde{d}_2)}. \tag{3.90}$$

With these two derivations we are capable of fusing two state estimates with closed-form solutions of two variants of CI.

3.5.5. Weighted Geometric Mean

So far we have considered the application of CI to two measurements. Most of the time this is sufficient for our use case. However, in Section 3.7.2 we will need to apply this concept to multiple measurements. To this end, we present a generalization of CI to multiple non-Gaussian probability density functions (PDFs) that is called WGM[6]. From this generalization we will recover the CI fusion rule for multiple Gaussian measure-

[6]Sometimes, it is also referred to as *Normalized Weighted Geometric Mean* (Bailey et al., 2012), *Weighted Geometric Density* (Julier, 2012), or *Generalized Uhlmann-Julier-Covariance Intersection* (Mahler, 2000).

ments as a special case.

The WGM fusion rule for N PDFs $p_i(X)$ is

$$p(X) = \frac{1}{\eta} \prod_{i=1}^{N} p_i(X)^{\omega_i} \tag{3.91}$$

with η begin a normalization constant, $\omega_i \geq 0$, and $\sum_i \omega_i = 1$. Note that we did not need to specify a type of distribution. For the particular case of Gaussian distributions the effect of raising the PDF to a power and normalizing it is the same as multiplying its covariance matrix with the inverse power. To see this, suppose that $\boldsymbol{X}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with PDF $p_i(X)$. Then, $\frac{1}{\eta_i} p_i(X)^{\omega_i}$ is equivalent to $\boldsymbol{X}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \frac{\boldsymbol{\Sigma}_i}{\omega_i})$. We use this result in combination with (3.91) to recover the CI fusion rule for N Gaussian measurements:

$$\boldsymbol{\Sigma}^{\omega} = \left(\sum_{i=1}^{N} \omega \boldsymbol{\Sigma}_i^{-1} \right)^{-1}, \tag{3.92}$$

$$\boldsymbol{x}^{\omega} = \boldsymbol{\Sigma}^{\omega} \left(\sum_{i=1}^{N} \omega \boldsymbol{\Sigma}_i^{-1} \boldsymbol{x}_i \right)^{-1}. \tag{3.93}$$

Again, $\omega_i \geq 0$ and $\sum_i \omega_i = 1$. Therefore, the detour of generalizing CI to WGM allows us to derive the CI fusion rule for multiple measurements.

3.6. Autoregressive models

Certain time-varying stochastic processes can be modeled as autoregressive models. Intuitively speaking, their output is influenced by their previous outputs plus additional stochastic noise. We denote an autoregressive model of order p as $AR(p)$. It is defined as

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t, \tag{3.94}$$

where the ϕ_i are the autocorrelation coefficients, c is a constant, and $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is a white noise process. The autocorrelation coefficients are also called the *parameters of the model*. With such a model we call the errors *autocorrelated* or *serially correlated*.

In later chapters we are interested in a particular kind of autoregressive model, the *zero mean autoregressive model of order 1 (AR(1)).* With $p = 1$ and $c = 0$ it is defined as

$$y_t = \phi y_{t-1} + \epsilon_t, \tag{3.95}$$

where we dropped the subscript on the autocorrelation coefficient ϕ_1.

Testing whether given data contains autocorrelated errors of order $p = 1$ can be accomplished by different test statistics, e.g., the Durbin-Watson test (Durbin and Watson, 1950). Let e_t be the residual for the t-th data point. The Durbin-Watson test statistic d is then

$$d = \frac{\sum_{t=2}^{T}(e_t - e_{t-1})^2}{\sum_{t=1}^{T} e_t^2}, \tag{3.96}$$

where T is the number of data points. The value of d is $d \approx 2(1 - \phi)$ and lies within the interval $[0, 4]$. It is compared to reference values to decide whether the errors are autocorrelated of order 1. It is close to $d \approx 2$ for $\phi = 0$.

The autocorrelation coefficients ϕ_i can be computed in multiple ways. These include the Yule-Walker equations (Yule, 1927), maximum likelihood estimation, or the Burg method (Brockwell et al., 2005). For our needs we content ourselves with inspecting the partial autocorrelation function of y_t and determining the model order and the autocorrelation coefficient.

Within a NLLSQ framework autocorrelated measurements can be treated with *generalized least squares* or *estimation in first differences*. The fields of statistics, econometrics, and financial mathematics have developed sophisticated tool sets for this, see for example the introduction by Maddala and Lahiri (1992). For our use case we are interested in time series of measurements whose noise is defined by an AR(1) model. We describe in Section 6.4 a method to incorporate pose estimates with autocorrelated errors.

3.7. Cramér-Rao Lower Bound and Fisher Information

In this section we present the concepts of the Cramér-Rao Lower Bound (CRLB) and the Fisher Information (FI). They will prove useful when computing the covariance of our estimated solution in the case of AR(1) models in Section 6.4.3. This derivation follows the Sections 2.1.8 and 2.2.11 in (Barfoot, 2017).

Assume that we want to estimate an unknown parameter $\boldsymbol{\theta}$ by observing a random variable X. The FI quantifies the information that a random variable X carries about a parameter $\boldsymbol{\theta}$ of a PDF that models X. We can express this relationship as the conditional PDF $p(X|\boldsymbol{\theta})$. The FI allows us to make statements about the best possible parameter estimation within this model. The FI matrix for an N-dimensional parameter space is defined as

$$\boldsymbol{I}(X|\boldsymbol{\theta}) = \mathbb{E}\left[\left(\frac{\partial \ln p(X|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)^\top \left(\frac{\partial \ln p(X|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)\right] \in \mathbb{R}^{N\times N}. \tag{3.97}$$

Suppose that we have obtained a set of realizations $\boldsymbol{x}$ of the random variable X. We can think of these realizations as a set of measurements. The CRLB states that the uncertainty of any unbiased estimate $\hat{\boldsymbol{\theta}}$ that is based on $\boldsymbol{x}$ has a lower bound. That is, given our measured information we cannot get more certain about our estimate than this lower bound. Now, the CRLB even allows us to compute this lower bound by stating that

$$\mathrm{cov}(\hat{\boldsymbol{\theta}}|\boldsymbol{x}) - \boldsymbol{I}^{-1}(X|\boldsymbol{\theta}) \geq 0. \tag{3.98}$$

Here the expression $\boldsymbol{A} \geq 0$ means that the matrix $\boldsymbol{A}$ is positive semi-definite[7]. Therefore, the CRLB fundamentally limits our certainty about the estimate of a parameter given a set of measurements. In other words, the measurements carry a limited amount of information for estimating a parameter. The FI quantifies this information.

3.7.1. Cramér-Rao Lower Bound applied to measurements from independent Gaussians

So far we have presented the relationship between the CRLB and the FI for general PDFs. We will now apply these concepts to measurements from statistically independent Gaussian PDFs.

Suppose we are interested in estimating the mean $\boldsymbol{\mu}$ of an N-dimensional Gaussian PDF. To this end, we obtain n measurements $\boldsymbol{x}_i \in \mathbb{R}^N$ of this distribution. The loga-

[7]If $\boldsymbol{A}$ expresses the difference between the estimated and the true covariance matrix, then this property is also called *covariance consistency*.

rithm of the joint probability of all measurements is

$$\ln p(X|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})^\top \boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu}) + \text{const}, \tag{3.99}$$

where $\boldsymbol{x} = [\boldsymbol{x}_1^\top, \dots, \boldsymbol{x}_n^\top]^\top$, $\boldsymbol{A} = \underbrace{[\mathbf{1}, \dots, \mathbf{1}]}_{n\text{ blocks}}^\top$, and $\boldsymbol{B} = \operatorname{diag}\underbrace{(\boldsymbol{\Sigma}, \dots, \boldsymbol{\Sigma})}_{n\text{ blocks}}$. For computing the CRLB we are interested in the derivative of (3.99) with respect to the estimated parameter $\boldsymbol{\mu}$:

$$\frac{\partial \ln p(X|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}} = (\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})^\top \boldsymbol{B}^{-1}\boldsymbol{A}. \tag{3.100}$$

The FI matrix is therefore

$$\begin{aligned}
\boldsymbol{I}(X|\boldsymbol{\mu}) &= \mathbb{E}\left[\left(\frac{\partial \ln p(X|\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}\right)^\top \left(\frac{\partial \ln p(X|\boldsymbol{\mu})}{\partial \boldsymbol{\mu}}\right)\right] && (3.101)\\
&= \mathbb{E}\left[\boldsymbol{A}^\top \boldsymbol{B}^{-1}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})^\top \boldsymbol{B}^{-1}\boldsymbol{A}\right] && (3.102)\\
&= \boldsymbol{A}^\top \boldsymbol{B}^{-1} \underbrace{\mathbb{E}\left[(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{\mu})^\top\right]}_{=\boldsymbol{B}} \boldsymbol{B}^{-1}\boldsymbol{A} && (3.103)\\
&= \boldsymbol{A}^\top \boldsymbol{B}^{-1}\boldsymbol{A} && (3.104)\\
&= n\boldsymbol{\Sigma}^{-1}. && (3.105)
\end{aligned}$$

Inserting this result into (3.98) allows us to compute the lower bound of the covariance of the estimated mean $\hat{\boldsymbol{\mu}}$ to

$$\operatorname{cov}(\hat{\boldsymbol{\mu}}|\boldsymbol{x}) \geq \frac{1}{n}\boldsymbol{\Sigma}. \tag{3.106}$$

Thus, we see that the uncertainty in the estimated mean shrinks with the number of measurements we have.

3.7.2. Cramér-Rao Lower Bound applied to Covariance Intersection

Until now we have investigated the general concept of the CRLB and also its application to measurements from independent Gaussians. Now we apply it in the context of CI. In this case, we are not assuming statistical independence between measurements anymore. However, the exact correlation between the measurements is unknown. Applying CI results in a covariance consistent estimate. If we examine the joint probability of the measurements, we obtain similar to (3.99)

$$\ln p(X|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{A\mu})^\top (\boldsymbol{B}^{\mathrm{CI}})^{-1}(\boldsymbol{x} - \boldsymbol{A\mu}) + \mathrm{const}, \tag{3.107}$$

but here $\boldsymbol{B}^{\mathrm{CI}} = \mathrm{diag}(\frac{1}{\omega_1}\boldsymbol{\Sigma}, \ldots, \frac{1}{\omega_n}\boldsymbol{\Sigma})$. Independently of how the weights ω_i are chosen we know that by construction $\sum_{i=1}^{n} \omega_i = 1$. Using this knowledge and (3.104) we conclude that the FI matrix for the scaled measurements is

$$\begin{aligned} \boldsymbol{I}^{\mathrm{CI}}(X|\boldsymbol{\mu}) &= \sum_{i=1}^{n} \frac{\boldsymbol{\Sigma}^{-1}}{\omega_i} &(3.108)\\ &= \boldsymbol{\Sigma}^{-1}. &(3.109)\end{aligned}$$

This allows us to compute the lower bound of the uncertainty for measurements that are fused with CI to

$$\mathrm{cov}(\hat{\boldsymbol{\mu}}|\boldsymbol{x}) \geq \boldsymbol{\Sigma}. \tag{3.110}$$

It is thus equal to the uncertainty of a *single* measurement. Therefore, the application of CI does not allow us to increase our certainty by obtaining more measurements. Instead, it aims to provide a covariance consistent estimate and has to implicitly assume that the information content of all measurements equals the information content of a single measurement. This is a conservative assumption that is based on not knowing anything about the correlation between the measurements. In case that we have more knowledge about it we will show in Section 6.4.3 how to exploit this knowledge to attain a lower bound of uncertainty.

4. Architecture of the pose fusion

We present in this chapter the overall architecture of our approach to the pose fusion. As we are treating pose sources about which we want to make as few assumptions as possible, we divide the main fusion task into two parts. The first part is the core estimator, which makes certain general error assumptions and performs the estimation. The second part consists of preprocessing techniques that increase the pose estimates' conformity with the error assumptions of the core estimator. They are applied to selected pose sources. In the following Section 4.1, we detail how these two parts are related to each other.

4.1. Layered fusion architecture

The architecture describes the fundamental partitioning of the pose fusion. This includes the main design decisions, the main components, and the interfaces between these.

Sensor fusion architectures can be categorized into the two major classes: *tightly* and *loosely* coupled approaches (Grewal et al., 2007; Steinhardt and Leinen, 2015; Hol, 2011). On the one hand, tightly coupled approaches directly use all sensor readings to compute the fused result. These approaches usually integrate the data at the position and velocity level. On the other hand, loosely coupled approaches rely on some form of preprocessing of the sensor readings before fusing them. These approaches usually integrate the data at the pseudorange, Doppler, or carrier phase level in the case of GNSS-based fusion (Eling, 2016). Both approaches span a range of fusion schemes in between that differ in the degree that they rely on certain forms of preprocessing. Figure 4.1 illustrates the key difference between these two architecture patterns for the example of GPS/IMU integration. This specific use case of GNSS/IMU integration is extensively reviewed in the literature. While some comparisons between tightly and

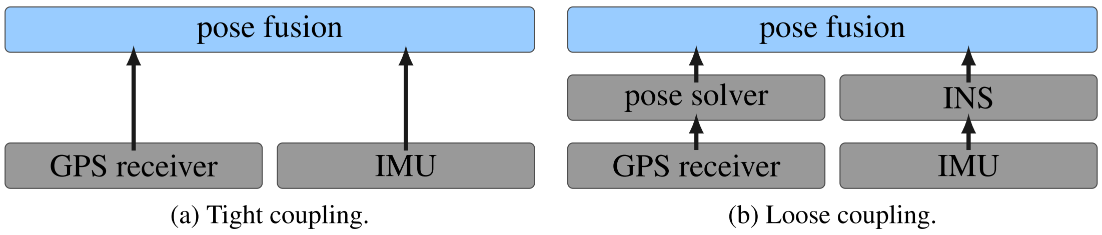


Figure 4.1.: Comparison of tightly and loosely coupled architectures for sensor fusion of data from a GPS receiver and an IMU.

loosely coupled approaches hint at a similar performance (Schwarz et al., 1994), there are others that show advantages for tightly coupled approaches (Scherzinger, 2000).

The architectural choice for one of these two patterns is straightforward for generic pose fusion. This is because its main challenge is that it cannot make specific assumptions about the sensors, concepts, or implementations of underlying pose sources. However, tightly coupled architectures are tailored to specific sensors and their measurements. Therefore, we build our generic pose fusion as a loosely coupled approach. This choice promotes the kind of modularity, flexibility, and extensibility that we want to gain from a generic pose fusion.

We approach the design of a loosely coupled system by proposing a layered fusion architecture, as shown in Figure 4.2. Layering is the organization of a system into separate functional components that interact in a hierarchical way. These functional components can be grouped to form a sublayer. Usually, each (sub)layer only has an interface to the layer below and above. Thus, layering is our main tool for complexity reduction and management.

4.2. Input, pose fusion, and application layer

Our layered fusion architecture consists of an input layer, a pose fusion layer, and an application layer.

Input layer The input layer contains the sensors and pose solvers. A combination of sensors and an appropriate sensor processing algorithm make up a *pose source*[1]. A

[1]Alternatively, we sometimes refer to it as *input source*.

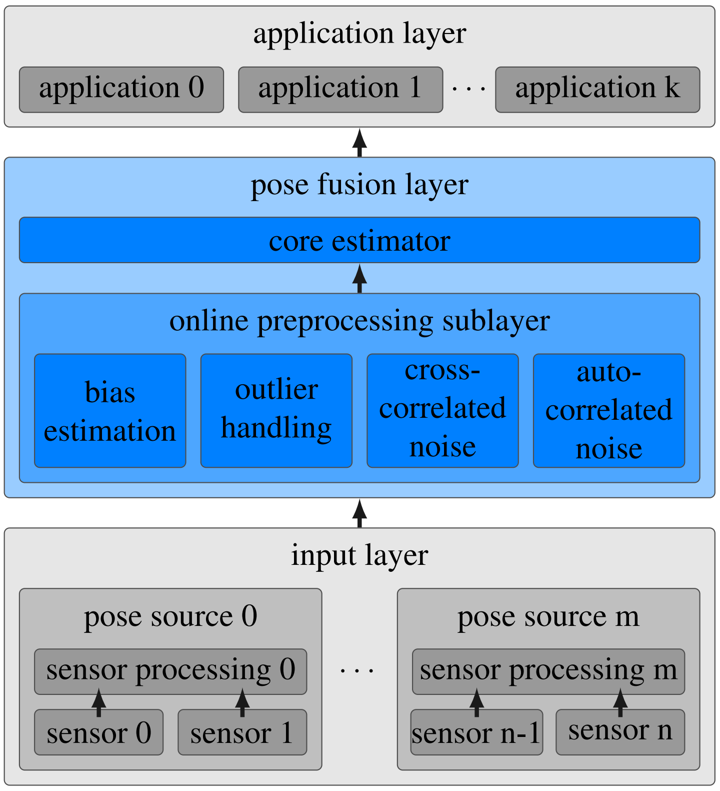


Figure 4.2.: Layer architecture of the pose fusion.

pose source builds upon a single or multiple sensors. Conversely, a sensor can feed into a single or into multiple pose sources. From the point of view of the pose fusion layer, the input layer serves to abstract from the specific sensor setup.

The interface between the input layer and the pose fusion layer is defined for information in the global or the vehicle reference frame. The i-th global pose estimate is given by the triple $(\boldsymbol{z}_i^{\mathrm{w}}, \boldsymbol{\Sigma}_i^{\mathrm{w}}, t_i)$ where $\boldsymbol{z}_i^{\mathrm{w}} = \left[x_i^{\mathrm{w}}, y_i^{\mathrm{w}}, \theta_i^{\mathrm{w}}\right]^{\top}$. $\boldsymbol{\Sigma}_i^{\mathrm{w}}$ is the pose estimate's covariance matrix. The time t_i specifies the time for which the pose estimate is valid. The i-th local pose estimate is similarly defined. It is given by the triple $(\boldsymbol{z}_i^{\mathrm{v}}, \boldsymbol{\Sigma}_i^{\mathrm{v}}, t_i)$ where $\boldsymbol{z}_i^{\mathrm{v}} = \left[\Delta x_i^{\mathrm{v}}, \Delta y_i^{\mathrm{v}}, \Delta\theta_i^{\mathrm{v}}\right]^{\top}$. The difference to global pose estimates is that local pose estimates encode a change of the current pose compared to the last pose rather than the pose itself. Again, $\boldsymbol{\Sigma}_i^{\mathrm{v}}$ is the covariance matrix of the pose estimate and t_i is its time. Generally, we refer to pose estimates transferred from the input to the pose fusion layer as *input pose estimates*.

Pose fusion layer The pose fusion layer receives pose estimates from the input layer. We subdivide this pose fusion layer into two modules: a sublayer of preprocessing techniques and the core estimator. The latter is the workhorse for the online state estimation. Its main purpose is to take a set of pose estimates, fuse them, and output an estimate of the current vehicle's pose. To this end, it has to make certain assumptions about the nature of the pose estimates' errors. In essence, it assumes that the error characteristics of the pose estimates can be modeled as AWGN. Even though this is a common assumption we can already provide for that these assumptions are not always met. Therefore, we need a set of preprocessing techniques.

We design the preprocessing sublayer such that we achieve modularity, flexibility, and extensibility. For this we create the preprocessing techniques souch that we can compose them in multiple ways. The best ordering of these techniques is dependent on the specific set of pose sources, as not every pose sources requires every preprocessing technique. Also, this sublayer is extensible in case that we want to preprocess pose sources in a different way. The set of modules in this thesis provides for a rich set of preprocessing that covers all relevant aspects of the pose sources that were available during the development of this thesis.

The preprocessing techniques manipulate the pose estimates such that they better fit the assumptions of the core estimator. In the spirit of generic pose fusion, we design these preprocessing techniques to be as universally applicable as possible. For this we classify the pose sources into sets that exhibit certain error characteristics. A preprocessing technique for one of these classes is then supposed to reduce this error characteristic as much as possible without making assumptions about the internal working of the pose sources. To clarify this structure, let us consider an example of an error characteristic. This could be a bias in the pose estimate of a global pose source. We group all pose sources that exhibit this characteristic into one class. It is hard to imagine a generic core estimator that produces a bias-free estimate if it relies on biased data. Therefore, one of our preprocessing techniques is to eliminate biases in global pose estimates as much as possible. We describe this approach in Section 6.1.

Another preprocessing technique allows us to better deal with outliers. We propose in Section 6.2 a method to adapt the covariance matrices of pose estimates based on prior knowledge. This prior knowledge is a precise map of the road. The effect is that the influence of certain pose estimates is downscaled.

The treatment of correlated errors between pose sources requires another preprocessing technique. We stated that a sensor can feed into a single or multiple pose sources. The latter case might lead to difficulties for the core estimator because it assumes that the errors of different pose sources are uncorrelated. This assumption might not hold if, for example, two pose sources rely on the same sensor. Generally, both pose sources will be affected by the identical sensor noise. Formally speaking, this leads to errors that are correlated between multiple pose sources. In Section 6.3 we will derive a method to treat this kind of difficulty.

Another challenge for the core estimator are pose estimates with autocorrelated errors. Suppose we have a pose source based on processing satellite signals. Errors due to multipath effects typically do not only affect a single pose estimate but rather a series of pose estimates. In this case, the core estimator's assumption of independently distributed pose estimates does not hold. Instead, it has to deal with autocorrelated errors. Therefore, we propose a preprocessing technique in Section 6.4 to reduce this effect.

The core estimator is at the heart of the pose fusion. Its main purpose is to take a set of pose estimates, fuse them, and output an estimate of the current vehicle's pose. Additionally, it provides the estimated uncertainty of this pose estimate. Formally, the i-th output of the core estimator—and with that, the output of the pose fusion layer—is the Gaussian distribution at time t_i with mean $\boldsymbol{x}_i$ and associated covariance matrix $\boldsymbol{\Sigma}_i$.

It is thus defined as the triple $(\boldsymbol{x}_i, \boldsymbol{\Sigma}_i, t_i)$. We call this the *output* or *fused pose estimate*.

Application layer The output of the pose fusion layer is passed to the application layer. Depending on the vehicle setup, there might be multiple applications that require a global pose estimate, e.g., path planning, routing, or collision avoidance. From the point of view of the application layer, the pose fusion layer acts like a centralized, virtual pose source. The pose sources are encapsulated behind the pose fusion layer. The applications do not directly interact with them and do not need to adjust for their different interfaces or behaviors. Instead, they receive a single unified pose estimate with its estimated uncertainty.

Software architecture We implement the pose fusion layer in the Automotive Data and Time-Triggered Framework (ADTF), see Appendix A for a brief introduction on this framework. Each module of the preprocessing sublayer corresponds to an ADTF *filter*. These filters are implemented in C++. The connections between the filters are easily adapted for a specific set of pose sources. The core estimator is implemented in a filter called *PoseGraphFusion (PGF).*

5. Core estimator

The core estimator is at the heart of the generic pose fusion[1]. It performs the state estimation given a set of preprocessed pose estimates. Figure 5.1 illustrates this problem for an example configuration in which two global and one local pose sources are available. We present in this chapter the concept of the core estimator in detail, which we call *PGF*. Our key contributions consist of an efficient sensor fusion algorithm that makes use of a graph-based formulation. We design the graph construction such that it produces a sparse block-tridiagonal structure in the system matrix. This can be solved quickly by graph optimization strategies. Furthermore, we embed the graph in a sliding window concept, where we show that marginalization can exactly and efficiently be carried out without an additional fill-in. The marginalization information can be represented by the novel concept of a prior node.

Within this chapter we derive in Section 5.1 the fundamental concepts behind the core estimator and relate it to other possible choices. Subsequently, we describe our approach to construct the main optimization problem in Section 5.2. This is extended in Section 5.3 where we detail our specific solution for performing and understanding marginalization. The timing characteristics of the core estimator are presented in Section 5.4. We exploit the knowledge about the runtime and timing requirements by proposing in Section 5.5 a resource-adaptive state estimation. In Section 5.6 we detail the last piece to the core estimator which is the assessment of the uncertainty of the fused pose estimate.

[1]We call it *core* estimator to highlight that the ensemble of preprocessing modules and estimator make up the entire pose estimation. Technically speaking, though, could we also refer to the core estimator simply as *estimator*.

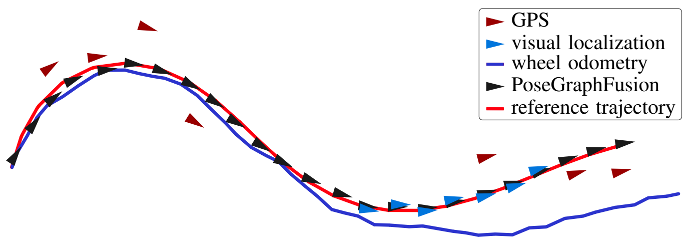


Figure 5.1.: Illustration of the state estimation problem based on a concrete example. Pose estimates from two global pose sources are available (a GPS receiver and a visual localization system) and from one local pose source (wheel odometry, shown as dead reckoning trajectory from the initial pose). The goal of the PGF is to estimate the reference trajectory (shown as red line) as closely as possible (its output is shown as black triangles).

5.1. Design of the core estimator

The core estimator is designed such that it satisfies the requirements we presented in Section 1.3. These demand an online estimator that works with three degrees of freedom. Moreover, it has to be configurable for different hardware setups and has to scale with the available computational power. In addition, we require a core estimator that accurately converges towards the statistically optimal result.

As detailed in Section 2.2, state estimation in the context of pose fusion is conventionally approached by using filtering-based approaches, such as the Kalman filter and its variants or, alternatively, smoothing algorithms. From a theoretical point of view, smoothing approaches benefit from the possibility to relinearize the Jacobians of all active state variables. They can be seen as a generalization of filtering-based approaches to multiple state variables. From a practical point of view, their problem formulation makes it easier to incorporate delayed or out-of-sequence pose estimates. This is important for a generic state estimator because multiple pose sources with different delays act in essence in the same way as out-of-sequence estimates. When all estimates from a pose source with a low delay have been processed, all later estimates from a pose source with a higher delay can be considered as out-of-sequence estimates for the current state. Smoothing algorithms are able to elegantly resolve time behavior issues and treat all sources in a homogeneous manner. For these reasons we design our core estimator as an optimization-based smoothing approach.

Smoothing approaches can be realized as either incremental or sliding window smoothing. For our approach we are not interested in an estimate over the full state at all times, which incremental smoothing aims to provide. Instead, it suffices to estimate the most recent pose. Furthermore, for incremental smoothing we have to keep the entire state in memory. This could potentially limit long-term operations without providing benefits. Therefore, we choose the general concept of a fixed-lag smoothing approach.

Motivated by the need for a powerful estimator and constrained by the requirement of an online solution, we design our core estimator with the same key concepts as a fixed-lag smoothing algorithm or a SWF. A general SWF as introduced by Sibley et al. (2010) usually also includes landmarks in its estimation process. In contrast to this, we do not include landmarks in the estimation process. This allows us to make adjustments in the graph construction phase specific to pose fusion that significantly influence the optimization. When compared to fixed-lag smoothing, we note that we are interested in the head of the lag (the most recent fused pose). Other than that, our approach is from a methodical point of view closest to fixed-lag smoothing and SWF approaches.

With this design choice, we inherit some of the main properties of a SWF. These include the accurate convergence towards the online batch estimation result, thereby providing the ML estimate. Furthermore, it is an efficient estimator that quickly reduces uncertainty and is consistent in the sense that it avoids overconfidence. Furthermore, it is a powerful property of SWFs that they scale from an IEKF to the online batch estimation by increasing the sliding window size. Putting things differently, we conclude that the IEKF and the online batch estimation are two special cases of a SWF. The SWF therefore generalizes these two concepts and is a suitable choice for our use case.

We choose to represent the SWF estimation with a pose graph. The motivation behind this is that we can represent all poses and their relations in a graphical way and perform inference over this graph to obtain the most likely set of hidden nodes. The graph-based representation provides intuition of how the state variables interact and how they are influenced by different pose estimates. It gives us a direct visual understanding of the sparsity pattern of the underlying NLLSQ problem. Furthermore, an arbitrary number of pose sources can be integrated into the graph in a modular way, where the information of one pose source does not influence the integration of another source's information into the graph. Also, the graph representation is robust against permanent and temporary failures of pose sources (or their sensors): if a pose source fails, this does not influence the graph construction phase for the other pose sources. This is different in data-driven

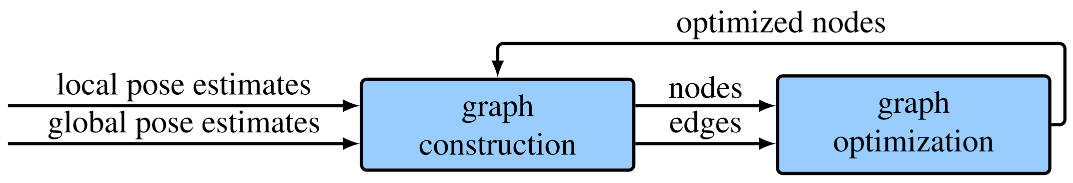


Figure 5.2.: Distinction between graph construction and optimization. This figure is inspired by Kümmerle (2013, Fig. 2.6).

approaches.

We split up the core estimator into two blocks, the graph construction and the graph optimization[2]. The first takes a set of local and global pose estimates and constructs the pose graph. It also takes care of marginalizing old nodes and sending the output to the application layer. The second block is the graph optimization. Its role is to optimize the pose graph and return the poses of the optimized nodes. The reason to split up the graph construction from its optimization is that we want to have the possibility to exchange the graph optimization if needed. Figure 5.2 illustrates the relation of the two concepts. Our implementation exploits the state-of-the-art graph optimization framework g2o (Kümmerle et al., 2011).

In the following we detail how we transform this general concept of the core estimator into a concrete approach. For this we are interested in formulating the NLLSQ approach for our use case. We derive this from the point of view of the graph construction and optimization.

5.2. Sliding window chain pose graphs

We design the core estimator as a sliding window NLLSQ estimator that is represented as a pose graph. In fact, by smoothing over a sliding window we effectively view pose fusion as trajectory estimation where we are specifically interested in the most recent pose of the trajectory. In this section we describe our graph construction approach that leads to the concept of sliding window chain pose graphs.

The input data to the core estimator are noisy odometry and global pose estimates.

[2]In graph-based SLAM, these two blocks are sometimes referred to as *frontend* and *backend* (Sünderhauf, 2012; Kümmerle, 2013).

Odometry sources provide the movement relative to the last pose, i.e., $\boldsymbol{z}_i^{\mathrm{v}} = \left[\Delta x_i^{\mathrm{v}}, \Delta y_i^{\mathrm{v}}, \Delta\theta_i^{\mathrm{v}}\right]^\top$ with the information matrix $\boldsymbol{\Lambda}_i^{\mathrm{v}}$. In contrast to odometry, global pose sources provide pose estimates in the global coordinate frame, i.e., $\boldsymbol{z}_i^{\mathrm{w}} = \left[x_i^{\mathrm{w}}, y_i^{\mathrm{w}}, \theta_i^{\mathrm{w}}\right]^\top$ with the information matrix $\boldsymbol{\Lambda}_i^{\mathrm{w}}$. Formally, we combine the set of all odometry estimates $\boldsymbol{z}^{\mathrm{v}} = \{\boldsymbol{z}_i^{\mathrm{v}}\}_{i=1}^{n^{\mathrm{v}}}$ with the set of all global pose estimates $\boldsymbol{z}^{\mathrm{w}} = \{\boldsymbol{z}_i^{\mathrm{w}}\}_{i=1}^{n^{\mathrm{w}}}$ to the set of all pose estimates $\boldsymbol{z} = \boldsymbol{z}^{\mathrm{v}} \cup \boldsymbol{z}^{\mathrm{w}}$.

In contrast to offline NLLSQ estimation, which is commonly taking into account all available information within the full pose graph, it is necessary for an online state estimation system to limit the considered information to keep the problem computationally tractable. Our approach achieves this by marginalizing out old state variables and thus only considering a fixed amount of state variables. More formally, in a sliding window pose graph the state vector $\boldsymbol{x}$ encompasses the M most recent state variables $\boldsymbol{x} = \left[\boldsymbol{x}_{t-M+1}^\top, \ldots, \boldsymbol{x}_t^\top\right]^\top$, where the state variable $\boldsymbol{x}_i = \left[x_i, y_i, \theta_i\right]$ is represented as a hidden pose. $\boldsymbol{x}_i$ is the i-th estimate of the vehicle's pose and is defined in the world reference frame. With this definition the size of the system matrix $\boldsymbol{H}$ is bounded by $\mathbb{R}^{3M\times 3M}$. The optimization result from the last time step provides the initial guess for the optimization in the current time step. This leads to an effective and efficient solution in practice. As most of the graph stays identical, a single optimization is usually sufficient to integrate the additional information of the current time step.

The key idea of the core estimator is that given the state vector $\boldsymbol{x}$ and the set of pose estimates $\boldsymbol{z}$ and their uncertainties, we seek the state $\boldsymbol{x}^*$ that best explains all pose estimates. To this end, we formulate a NLLSQ problem with these quantities as described in Section 3.4. More formally, we seek the state $\boldsymbol{x}^*$ such that

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \sum_{i=1}^{n} \boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i)^\top \boldsymbol{\Lambda}_i \boldsymbol{e}_i(\boldsymbol{x}, \boldsymbol{z}_i) \tag{5.1}$$

$$= \arg\min_{\boldsymbol{x}} \underbrace{\sum_{i=1}^{n^{\mathrm{v}}} \boldsymbol{e}_i^{\mathrm{v}}(\boldsymbol{x}, \boldsymbol{z}_i^{\mathrm{v}})^\top \boldsymbol{\Lambda}_i^{\mathrm{v}} \boldsymbol{e}_i^{\mathrm{v}}(\boldsymbol{x}, \boldsymbol{z}_i^{\mathrm{v}})}_{\text{odometry constraints}} + \underbrace{\sum_{i=1}^{n^{\mathrm{w}}} \boldsymbol{e}_i^{\mathrm{w}}(\boldsymbol{x}, \boldsymbol{z}_i^{\mathrm{w}})^\top \boldsymbol{\Lambda}_i^{\mathrm{w}} \boldsymbol{e}_i^{\mathrm{w}}(\boldsymbol{x}, \boldsymbol{z}_i^{\mathrm{w}})}_{\text{global pose constraints}}. \tag{5.2}$$

In this formulation we split up the global error function $F(\boldsymbol{x})$ into the first term, which sums up all odometry constraints, and the second term, which sums up all global pose constraints. This allows us to treat these constraints, their error functions, and their

Jacobians separately. Note that we use the entire state vector $\boldsymbol{x}$ as argument for both odometry and global pose error functions for notational convenience even though that $\boldsymbol{e}_i^{\mathrm{v}}(\boldsymbol{x}, \boldsymbol{z}_i^{\mathrm{v}})$ depends only on two and $\boldsymbol{e}_i^{\mathrm{w}}(\boldsymbol{x}, \boldsymbol{z}_i^{\mathrm{w}})$ only on a single state variable.

We can compute $\boldsymbol{H}$ and $\boldsymbol{b}$ as described in Section 3.4.2 by choosing the correct error functions and Jacobians for each constraint. One important question that we have not yet addressed is how to derive the constraints from the input pose estimates. We implied until now that we can derive one constraint from one input pose estimate. However, these cannot be mapped one-to-one into the same pose graph if input pose estimates arrive with unknown, fluctuating, and nonsynchronized rates. The rationale behind this is best illustrated with an example. Consider a global and a local pose source that provide nonsynchronized pose estimates, i.e., the pose estimates of one source are defined for different time steps than those of the other source. If we assume for a moment a one-to-one correspondence of pose estimates to constraints, then we start by adding observed and hidden nodes for all pose estimates of the global pose source. However, the local pose estimates will not lead to constraints between these hidden nodes as they are valid at different time steps. Even if we introduce new hidden nodes for the local pose estimates, then we can still not connect them to the hidden nodes that were induced by the global pose estimates.

Therefore, any graph-based system of this kind needs to develop a strategy to handle this challenge. Our approach is to interpolate the pose estimates to the same time steps. What seems like a minor implementation problem actually influences the estimation quality: first, it generally introduces an interpolation error. Secondly, the interpolation of pose estimates leads to correlated constraints. The magnitude of correlation depends on the exact kind of interpolation. The core estimator however assumes statistically independent constraints. Therefore, we keep the interpolations to a minimum by adding hidden nodes so that the interpolations are small in a temporal sense. This leads us to the question of when to create hidden and observed nodes.

In the following we describe our graph construction concept that leads to the definition of chain pose graphs. We design it such that it enforces a matrix structure for $\boldsymbol{H}$ that is particularly easy to solve, includes all pose sources in a generic way independently of their specific output frequencies, and a priori relates the number of state variables to the length of the interval of the sliding window. The necessary steps for this are the description of a time-triggered hidden node construction, the integration of observed nodes, and finally the integration of odometry constraints.

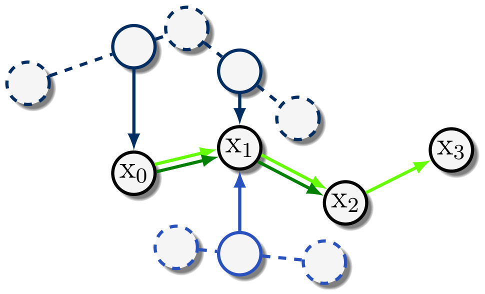
In contrast to related graph-based approaches (Cucci and Matteucci, 2013; Sibley et al., 2010), we neither generate a hidden node every time a pose estimate arrives nor tie its generation to a specific pose source (so-called data-triggered approaches). Instead, we construct a hidden node every time step (time-triggered approach), i.e., with the *temporal resolution* Δt. This is advantageous because the graph construction does not fail as in data-triggered approaches if this specific pose source fails to provide pose estimates. Moreover, we can define the interval of the sliding window without considering the output frequency of this specific pose source. Also, we can control Δt such that the interpolation of pose estimates does not severely impact the estimation quality.

Having decided when hidden nodes are constructed, it is straightforward to define how observed nodes are integrated. Ideally, we create one observed node per global pose estimate, but generally, the timestamps will not match exactly. In this case, we construct one observed node per global pose estimate by interpolating two successive global pose estimates of the same source. The interpolation has to take into account the law of propagation of uncertainty as detailed by Schlegel et al. (2012).

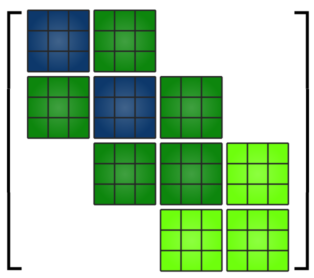
The integration of odometry information into the pose graph can be distinguished into two different approaches. A straightforward way consists in adding edges such that the timestamps of corresponding nodes conform as closely as possible to the timestamps of the pose estimates. This is a temporal nearest neighbor approach which ignores the synchronization difficulty. We do not follow this approach as it means that edges can potentially connect hidden nodes that are not adjacent. This in turn results in an unrestricted graph structure because with a growing number of odometry sources, any two hidden nodes can potentially be connected. We will detail the downsides of this approach after describing ours.

Instead of following the strategy just described, we propose to query each odometry source to interpolate the edges between each two successive hidden nodes. We refer to the resulting form of the graph as *chain pose graph*. Constructing edges from odometry estimates only between successive hidden nodes via interpolation is not merely an implementation detail but is in fact crucial as there are significant gains in terms of runtime and memory efficiency by using chain pose graphs. We derive these by analyzing how the matrix structure of $\boldsymbol{H}$ is defined by our chain pose graph construction.

The block structure of $\boldsymbol{H}$ reflects the connections of poses and edges in the graph



(a) A chain pose graph.



(b) Corresponding block-tridiagonal structure of the system matrix.

Figure 5.3.: A chain pose graph and the corresponding structure of the system matrix. a) The black circles are hidden nodes, the dashed blue circles are global pose estimates from two different sources, the non-dashed blue circles are observed nodes, and the green edges are odometry constraints. Note how the global pose estimates are interpolated (dashed blue lines) at the timestamps of the hidden nodes to obtain the observed nodes. b) The structure of $\boldsymbol{H}$ for the graph in (a). The blue block entries are caused by the observed nodes and the odometry constraints. The dark green block entries are caused by the four odometry constraints between x_0, x_1, and x_2. The light green block entries are caused by the odometry constraint between x_2 and x_3.

as it is its adjacency matrix. Its structure changes slightly with the availability of constraints. In general, the block structure of a chain pose graph is a block-tridiagonal matrix. The example graph in Figure 5.3 illustrates how to integrate multiple hidden and observed nodes as well as odometry constraints. It additionally shows the resulting block-tridiagonal matrix structure of the corresponding system matrix. The diagonal entries in the system matrix are influenced by odometry and global pose constraints while the off-diagonal entries are only affected by odometry constraints. Loop closures are not considered as they arise rarely when driving straight from destination to target.

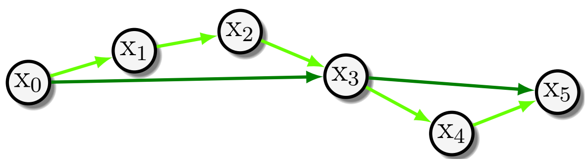
The block-tridiagonal structure is a consequence of the linear temporal ordering of the state variables combined with the fact that odometry constraints are constructed only between successive hidden nodes. We specifically design our solution to produce a block-tridiagonal matrix because this structure has multiple advantages. First, it does not produce fill-in in $\boldsymbol{H}$ after marginalization of the oldest state variables. This effect occurs in non-chain pose graph structures because marginalization summarizes the information of the eliminated node in edges between all pairs of variables that are directly related to the eliminated node (Eustice et al., 2006a; Kretzschmar et al., 2011). These newly introduced constraints capture the marginalized information in the so-called elimination clique, leading to a denser system matrix.

Secondly, even the Cholesky factorization $\boldsymbol{H} = \boldsymbol{R}^\top \boldsymbol{R}$ which we perform to solve the linear system does not suffer from fill-in in its triangular matrix $\boldsymbol{R}$. In fact, $\boldsymbol{R}$ becomes a band matrix as illustrated in Figure 5.6. As a consequence, costly variable reordering techniques (Agarwal and Olson, 2012) are unnecessary as $\boldsymbol{R}$ already contains the minimum number of nonzero elements necessary to reconstruct $\boldsymbol{H}$. Figure 5.4 and Figure 5.5 visualize the different effects for factorization and marginalization of a non-chain pose graph compared to a chain pose graph. By comparing them it becomes apparent that $\boldsymbol{H}$ and $\boldsymbol{R}$ have less nonzero elements when constructing a chain pose graph. Moreover, $\boldsymbol{H}$ does not suffer from fill-in after marginalization.

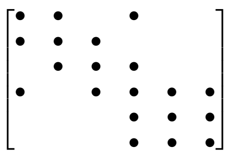
Thirdly, the computational complexity of the Cholesky factorization of a block-tridiagonal matrix is $\mathcal{O}(n)$ (with n being the number of nonzero entries in $\boldsymbol{H}$). This is a substantial improvement compared to arbitrary (dense) matrix structures whose decomposition or inversion becomes as costly as approximately $\mathcal{O}(n^{2.4})$. For experiments concerning the runtime for general graph-based optimization with g2o, we refer the reader to the runtime evaluations by Kümmerle et al. (2011, Fig. 8). There, the time complexity for optimizing the Manhattan3500 dataset (Olson et al., 2006) is clearly higher than linear in the number of nodes although the same sparse matrix techniques as in our implementation were used. For our approach, we demonstrate the linear time complexity for a practical experiment in the evaluations section.

In summary, our chain pose graph approach prevents fill-in after marginalization in $\boldsymbol{H}$ and during the factorization in $\boldsymbol{R}$, makes common variable reordering strategies unnecessary (thus speeding up computation), and is efficiently solvable in $\mathcal{O}(n)$. The memory consumption for both $\boldsymbol{H}$ and $\boldsymbol{R}$ remains constant over time as both matrices are bounded in size. Also, the optimization problem remains efficiently solvable in $\mathcal{O}(n)$. In total, the time and memory complexities of our approach are linear in the number of hidden nodes and independent of the duration of operation because of the chain pose graph structure.

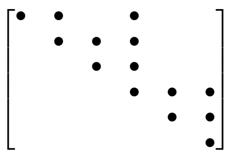
So far we have detailed the graph structure but merely touched the topic of keeping them in a sliding window. In the following we explain this by detailing the marginalization process.



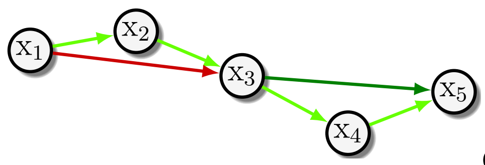
(a) An example graph that is not a chain pose graph.



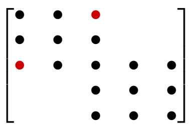
(b) Sparsity pattern of the system matrix $\boldsymbol{H}$ for the graph in (a).



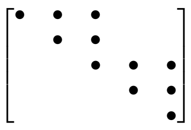
(c) Sparsity pattern of the Choleksy factorization $\boldsymbol{R}$ for the matrix in (b).



(d) Graph structure after marginalizing x_0. A new edge due to the marginalization is displayed in red.

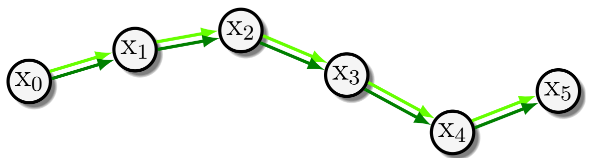


(e) Sparsity pattern of the system matrix $\boldsymbol{H}$ for the graph in (d). Fill-in due to the marginalization is marked in red.

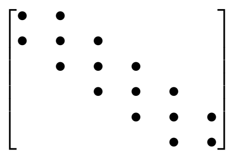


(f) Sparsity pattern of the Cholesky factorization $\boldsymbol{R}$ for the matrix in (e).

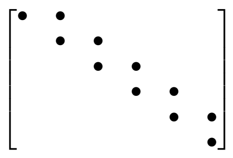
Figure 5.4.: Factorization and marginalization of a non-chain pose graph. Marginalizing x_0 leads to a new edge in the graph (namely from x_1 to x_3) and thus to a denser system matrix. This reduces its sparsity and consequently decreases runtime performance. Also, it is potentially necessary to apply a variable reordering strategy to reduce the number of nonzero elements in $\boldsymbol{R}$.



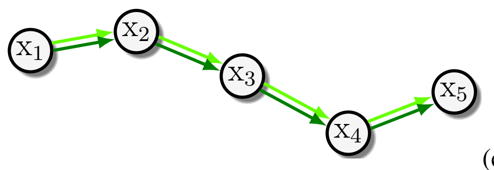
(a) A chain pose graph. Odometry estimates are broken down such that they create edges between adjacent hidden nodes.



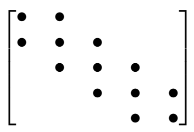
(b) Sparsity pattern of the system matrix $\boldsymbol{H}$ for the graph in (a).



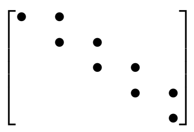
(c) Sparsity pattern of the Cholesky factorization $\boldsymbol{R}$ for the matrix in (b).



(d) Graph structure after marginalizing x_0. The marginalization does not result in new edges.

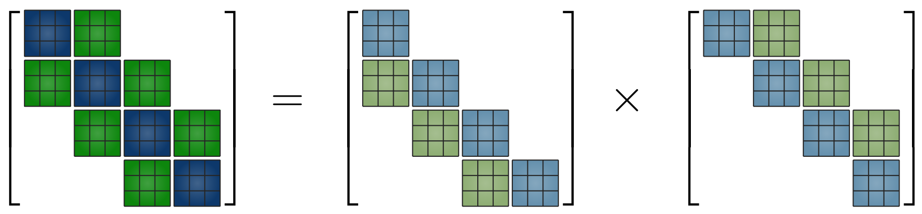


(e) Sparsity pattern of $\boldsymbol{H}$ for the graph in (d). The marginalization does not result in fill-in.



(f) Sparsity pattern of the Cholesky factorization $\boldsymbol{R}$ for the matrix in (e).

Figure 5.5.: Factorization and marginalization of a chain pose graph. Marginalizing x_0 does not lead to new edges in the graph or fill-in in the system matrix. Variable reordering strategies are unnecessary as $\boldsymbol{R}$ contains the minimum number of nonzero elements necessary to reconstruct $\boldsymbol{H}$.



(a) Block-tridiagonal structure of the system matrix $\boldsymbol{H}$.

(b) Block structure of the corresponding Cholesky matrix $\boldsymbol{R}^\top$.

(c) Block structure of $\boldsymbol{R}$.

Figure 5.6.: Block structure of $\boldsymbol{H}$ and its Cholesky decomposition $\boldsymbol{R}^\top\boldsymbol{R}$.

5.3. Marginalization as a prior node

We have detailed the chain pose graph construction in Section 5.2 where we stated that we marginalize the oldest state variables to achieve a sliding window effect. We complete this by analyzing our specific marginalization method in this section. The key insight here is that we can understand marginalization in terms of a special observed node which we will name *prior node*.

We have already seen that it is mandatory to limit the number of hidden nodes to keep the problem computationally tractable. Simply removing edges and nodes leads to information loss and is equivalent to conditioning, which potentially leads to overconfidence. Therefore, we marginalize the oldest nodes. Marginalization produces the marginal probability distribution over the subset of involved nodes and retains the information about how they interact. It truncates the graph by discarding the state variables that are being marginalized out but retains the same information for the remaining state variables (given the linearization point).

However, excluding nodes from the sliding window always has the drawback that their linearization point for the NLLSQ optimization is fixed. It is the powerful ability of relinearization that leads to a high estimation quality for fixed-lag smoothing approaches, as we discussed in Section 2.2. Marginalizing hidden nodes naturally results in the situation that the estimates of excluded nodes cannot be adjusted anymore. Consequently, it is crucial to only marginalize "mature" nodes with a converged estimate. The proposed marginalization method accounts for this by leaving all hidden nodes as long as possible in the graph and eliminating only the oldest ones.

Marginalization can be carried out approximately or exactly. As we have considered the impact of exact marginalization in the definition of chain pose graphs, we can apply the exact solution. The common approach for exact marginalization is computing the Schur complement on the system matrix $\boldsymbol{H}$. In general, the disadvantage of this operation is the introduction of conditional dependencies between state variables that are connected. As we design our problem structure to be a chain pose graph, we are able to retain the same sparsity pattern and do not suffer from a denser system matrix after marginalization as we show in Section 5.2.

We closely examine the effect of the Schur complement on chain pose graphs and observe that only the top left block entry of $\boldsymbol{H}$ changes, see Figure 5.7. Interestingly,

$$\begin{bmatrix} \boldsymbol{H}_{11} & \boldsymbol{H}_{12} & & & \\ \boldsymbol{H}_{12}^{\top} & \boldsymbol{H}_{22} & \boldsymbol{H}_{23} & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{H}_{n-2n-1}^{\top} & \boldsymbol{H}_{n-1n-1} & \boldsymbol{H}_{n-1n} \\ & & & \boldsymbol{H}_{n-1n}^{\top} & \boldsymbol{H}_{nn} \end{bmatrix}$$

(a) System matrix $\boldsymbol{H}$ before marginalization.

$$\begin{bmatrix} \tilde{\boldsymbol{H}}_{22} & \boldsymbol{H}_{23} & & & \\ \boldsymbol{H}_{23}^{\top} & \boldsymbol{H}_{33} & \boldsymbol{H}_{34} & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{H}_{n-2n-1}^{\top} & \boldsymbol{H}_{n-1n-1} & \boldsymbol{H}_{n-1n} \\ & & & \boldsymbol{H}_{n-1n}^{\top} & \boldsymbol{H}_{nn} \end{bmatrix}$$

(b) $\boldsymbol{H}$ after marginalization of the oldest state variable. Note that the entry $\boldsymbol{H}_{22}$ has changed as it contains information of the marginalized state variable.

Figure 5.7.: The effect of marginalization on the system matrix $\boldsymbol{H}$. Only the block entry $\boldsymbol{H}_{22}$ changes. This motivates us to derive the concept of a prior node.

adding an observed node to the graph has a similar effect: only the block entry of the corresponding hidden node on the diagonal of $\boldsymbol{H}$ gets an additional addend. Combining these two facts motivates us to study whether the effect of marginalization on a chain pose graph can be represented by an additional observed node.

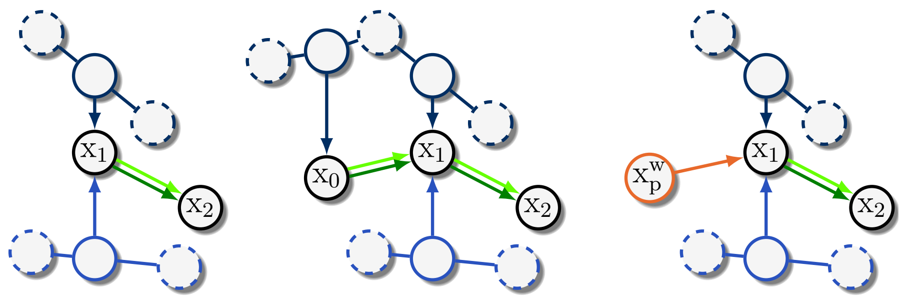
Therefore, we show that we can exploit the knowledge of the particular block-tridiagonal matrix structure to derive the concept of a *prior node* which carries the same information as introduced by the Schur complement. In general, using a representation in the form of a graph is beneficial compared to directly solving the NLLSQ equations because of the possibility to visually understand the relations of the state variables, more possibilities for data inspection, and a more intuitive way to manipulate the problem structure. These are the same reasons why it is advantageous to construct a prior node for marginalization instead of performing the Schur complement. The user has the possibility to understand how the prior information affects the rest of the graph, thus allowing him to manipulate this information if desired. If one was to repeatedly perform the Schur complement, it would become untraceable to understand the optimization result of the graph as not all necessary information is conceptually represented herein. The concept of a prior node is also supported by a more pragmatic reason: it allows us to store and load the optimization problem with solely the help of its graph representation. Furthermore, it opens up the possibility to explicitly apply a robust cost function on the prior node constraint and to adjust the uncertainty of the prior information based on context. In total, marginalization by using a prior node is making the graph construction logic aware of the marginalization process, allows the understanding and manipulation of the prior information, and is thus preferable over marginalization by performing the Schur complement on the system matrix.

In the following we will first analyze the impact of the Schur complement on our chain pose graphs and subsequently show how to compute the uncertainty and mean estimate of the prior node to obtain the same result. The derivation is detailed for the marginalization of a single hidden node but can easily be applied iteratively if multiple nodes shall be marginalized. As a result, we obtain the prior node that has the same effect as the Schur complement and therefore represents the effect of marginalization with the help of graph elements.

5.3.1. Exact marginalization with the Schur complement

For general pose graphs performing the Schur complement leads to the exact and consistent marginalization of nodes. This results in challenges as it generally leads to the introduction of new conditional dependencies between hidden nodes. First, these are not always trivial to compute. Secondly, these make the graph and its system matrix more dense, therefore causing a higher solution time. Chain pose graphs circumvent these challenges by design. In the following we restrict ourselves to the study of the marginalization of the oldest hidden node in a chain pose graph. We will show that this computation is straightforward and that it does not influence the sparsity pattern of $\boldsymbol{H}$.

It is useful for the derivation to start with a graph with only two hidden nodes (see Figure 5.8a) and ignore the node x_0 for now to see how the different terms are affected by the marginalization. Consider a graph $\overrightarrow{\mathcal{G}}^{\mathrm{small}}$ where x_1 is linked to x_2 and additionally to one or more observed nodes (Figure 5.8a shows an example for two connected observed nodes, depicted by blue circles). The corresponding system matrix $\boldsymbol{H}^{\mathrm{small}}$ is given as a



(a) Graph $\overrightarrow{\mathcal{G}}^{\text{small}}$. (b) Graph $\overrightarrow{\mathcal{G}}^{\text{full}}$. (c) Graph $\overrightarrow{\mathcal{G}}^{\text{prior}}$.

Figure 5.8.: Graph marginalization can be understood as prepending a prior node x_p^w to the graph. Blue non-dashed circles represent observed nodes. For understanding how the hidden node x_0 influences the optimization it is useful to start with a graph in (a) without x_0. Considering x_0 in (b) leads to additional terms in the system matrix. Marginalization in (c) with a prior node x_p^w leads to the same system matrix as performing the conventional Schur complement on $\overrightarrow{\mathcal{G}}^{\text{full}}$.

block matrix and the optimization solves the equation

$$\begin{bmatrix} \boldsymbol{H}_{11}^{\text{small}} & \boldsymbol{H}_{12}^{\text{small}} \\ \boldsymbol{H}_{21}^{\text{small}} & \boldsymbol{H}_{22}^{\text{small}} \end{bmatrix} \Delta \boldsymbol{x}^{\text{small}} = - \begin{bmatrix} \boldsymbol{b}_1^{\text{small}} \\ \boldsymbol{b}_2^{\text{small}} \end{bmatrix}. \tag{5.3}$$

If we additionally include the hidden node x_0, the graph structure and the system matrix change. Let x_0 also be connected to one or more observed nodes and consider one or more odometry constraints between x_0 and x_1. The resulting graph $\overrightarrow{\mathcal{G}}^{\text{full}}$ (see Figure 5.8b) is defined by the system matrix

$$\boldsymbol{H}^{\text{full}} = \begin{bmatrix} \boldsymbol{H}_{00}^{\text{w}} + \boldsymbol{H}_{00}^{\text{v}} & \boldsymbol{H}_{01}^{\text{v}} & \\ \boldsymbol{H}_{10}^{\text{v}} & \boldsymbol{H}_{11}^{\text{small}} + \boldsymbol{H}_{11}^{\text{v}} & \boldsymbol{H}_{12}^{\text{small}} \\ & \boldsymbol{H}_{21}^{\text{small}} & \boldsymbol{H}_{22}^{\text{small}} \end{bmatrix} \tag{5.4}$$

and the coefficient vector

$$\boldsymbol{b}^{\text{full}} = \begin{bmatrix} \boldsymbol{b}_0^{\text{full}} \\ \boldsymbol{b}_1^{\text{full}} \\ \boldsymbol{b}_2^{\text{full}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0^{\text{w}} + \boldsymbol{b}_0^{\text{v}} \\ \boldsymbol{b}_1^{\text{small}} + \boldsymbol{b}_1^{\text{v}} \\ \boldsymbol{b}_2^{\text{small}} \end{bmatrix}, \tag{5.5}$$

where $\boldsymbol{H}_{ij}^{\mathrm{w}}$ and $\boldsymbol{H}_{ij}^{\mathrm{v}}$ capture the sum of all entries of $\boldsymbol{H}_{ij}$ stemming from observed nodes and odometry constraints, respectively. The terms $\boldsymbol{b}_i^{\mathrm{w}}$ and $\boldsymbol{b}_i^{\mathrm{v}}$ have the same function for the vector entry $\boldsymbol{b}_i$. All of these terms are readily available as they have been computed in the last iteration, including any robust cost function applied on them.

The common way to marginalize x_0 consists in computing the Schur complement of $\boldsymbol{H}^{\mathrm{full}}$ and $\boldsymbol{b}^{\mathrm{full}}$. This leads to the marginalized system matrix $\boldsymbol{H}^{\mathrm{marg}}$, which is identical to $\boldsymbol{H}^{\mathrm{small}}$ except for the upper left block which changes to

$$\boldsymbol{H}_{11}^{\mathrm{marg}} = \boldsymbol{H}_{11}^{\mathrm{small}} + \boldsymbol{H}_{\mathrm{schur}}, \tag{5.6}$$

$$\boldsymbol{H}_{\mathrm{schur}} = \boldsymbol{H}_{11}^{\mathrm{v}} - \boldsymbol{H}_{10}^{\mathrm{v}}(\boldsymbol{H}_{00}^{\mathrm{w}} + \boldsymbol{H}_{00}^{\mathrm{v}})^{-1}\boldsymbol{H}_{01}^{\mathrm{v}}. \tag{5.7}$$

The corresponding marginalized coefficient vector is

$$\boldsymbol{b}^{\mathrm{marg}} = \begin{bmatrix} \boldsymbol{b}_1^{\mathrm{small}} + \boldsymbol{b}_{\mathrm{schur}} \\ \boldsymbol{b}_2^{\mathrm{small}} \end{bmatrix}, \tag{5.8}$$

$$\boldsymbol{b}_{\mathrm{schur}} = \boldsymbol{b}_1^{\mathrm{v}} - \boldsymbol{H}_{10}^{\mathrm{v}}(\boldsymbol{H}_{00}^{\mathrm{w}} + \boldsymbol{H}_{00}^{\mathrm{v}})^{-1}(\boldsymbol{b}_0^{\mathrm{w}} + \boldsymbol{b}_0^{\mathrm{v}}). \tag{5.9}$$

We remark that the structure of the system matrix is still block-tridiagonal after the marginalization due to the particular design of our chain pose graph. This means that the sparsity pattern of $\boldsymbol{H}$ is retained after marginalization without fill-in. Additionally, $\boldsymbol{H}_{\mathrm{schur}}$ and $\boldsymbol{b}_{\mathrm{schur}}$ are easily computable as the inversion for $(\boldsymbol{H}_{00}^{\mathrm{w}} + \boldsymbol{H}_{00}^{\mathrm{v}})^{-1}$ is only needed over a 3×3 matrix. Moreover, the information is conserved in a consistent way. Any other marginalization method that claims to be exact needs to produce the same result. In the remainder of this section we show that an alternative marginalization technique consists in replacing x_0 and its connected observed nodes and edges with a prior node $\mathrm{x}_{\mathrm{p}}^{\mathrm{w}}$ which behaves like an observed node. To this end, two questions have to be answered: how to compute its correct mean estimate and how to compute the uncertainty of this prior node?

5.3.2. Closed-form solution of the prior node

We detailed in Section 5.3.1 that exact marginalization does not result in fill-in in the system matrix for chain pose graphs and derived how the Schur complement affects the upper left part of $\boldsymbol{H}$ and $\boldsymbol{b}$. Our goal is now to devlop the equations for computing the prior node $\mathrm{x}_\mathrm{p}^\mathrm{w}$, which is depicted in Figure 5.8c. First of all we observe that in the special cases that x_0 is not directly connected to any observed node or that there exist no edges between x_0 and x_1, x_0 does not influence the estimate of x_1 and we can omit constructing a prior node. The derivations in the following treat the general case in which x_0 is connected to at least one observed node (which can for example be the prior node from the last cycle) and is additionally linked via at least one edge to x_1.

The first step is to remove x_0 as well as all observed nodes and edges connected to it. Connecting the prior node $\mathrm{x}_\mathrm{p}^\mathrm{w}$ with the information matrix $\boldsymbol{\Lambda}_\mathrm{p}^\mathrm{w}$ to x_1 leads to the graph $\overrightarrow{\mathcal{G}}^\mathrm{prior}$ (see Figure 5.8c). The prepended node yields an additional addend $\boldsymbol{H}_\mathrm{p}^\mathrm{w}$ in the matrix $\boldsymbol{H}^\mathrm{prior}$ of $\overrightarrow{\mathcal{G}}^\mathrm{prior}$ such that

$$\boldsymbol{H}^\mathrm{prior} = \begin{bmatrix} \boldsymbol{H}_{11}^\mathrm{small} + \boldsymbol{H}_\mathrm{p}^\mathrm{w} & \boldsymbol{H}_{12}^\mathrm{small} \\ \boldsymbol{H}_{21}^\mathrm{small} & \boldsymbol{H}_{22}^\mathrm{small} \end{bmatrix}, \tag{5.10}$$

and an additional addend $\boldsymbol{b}_\mathrm{p}^\mathrm{w}$ in $\boldsymbol{b}^\mathrm{prior}$ such that

$$\boldsymbol{b}^\mathrm{prior} = \begin{bmatrix} \boldsymbol{b}_1^\mathrm{small} + \boldsymbol{b}_\mathrm{p}^\mathrm{w} \\ \boldsymbol{b}_2^\mathrm{small} \end{bmatrix}. \tag{5.11}$$

We want to create the prior node so that it behaves like any other observed node in the graph. This already determines the error function and its Jacobian for the constraint induced by the prior node. Let $\boldsymbol{e}_\mathrm{p} = \boldsymbol{e}^\mathrm{w}(\mathrm{x}_1, \boldsymbol{x}_\mathrm{p}^\mathrm{w})$ be the error function and $\breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x})$ its Jacobian evaluated at the current linearization point. We recall that (cf. (3.52) and (3.53))

$$\boldsymbol{e}_\mathrm{p} = \begin{bmatrix} \boldsymbol{R}_{\theta_\mathrm{p}}^\top & 0 \\ \boldsymbol{0}^\top & 1 \end{bmatrix} (\boldsymbol{x}_1 - \boldsymbol{x}_\mathrm{p}^\mathrm{w}), \tag{5.12}$$

$$\breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{R}_{\theta_\mathrm{p}}^\top & 0 \\ \boldsymbol{0}^\top & 1 \end{bmatrix}. \tag{5.13}$$

We need these two terms to compute $\boldsymbol{H}_\mathrm{p}^\mathrm{w}$ and $\boldsymbol{b}_\mathrm{p}^\mathrm{w}$:

$$\boldsymbol{H}_\mathrm{p}^\mathrm{w} = \breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x})^\top \boldsymbol{\Lambda}_\mathrm{p}^\mathrm{w} \breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x}), \tag{5.14}$$

$$\boldsymbol{b}_\mathrm{p}^\mathrm{w} = \breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x})^\top \boldsymbol{\Lambda}_\mathrm{p}^\mathrm{w} \breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}_\mathrm{p}^\mathrm{w}). \tag{5.15}$$

With this insight we derived how $\mathrm{x}_\mathrm{p}^\mathrm{w}$ influences $\boldsymbol{H}^\mathrm{prior}$, $\boldsymbol{b}^\mathrm{prior}$ through $\boldsymbol{H}_\mathrm{p}^\mathrm{w}$, $\boldsymbol{b}_\mathrm{p}^\mathrm{w}$. We have also shown that the Schur complement influences $\boldsymbol{H}^\mathrm{marg}$, $\boldsymbol{b}^\mathrm{marg}$ through $\boldsymbol{H}_\mathrm{schur}$, $\boldsymbol{b}_\mathrm{schur}$. The last step consists in proving $\boldsymbol{H}_\mathrm{p}^\mathrm{w} = \boldsymbol{H}_\mathrm{schur}$ and $\boldsymbol{b}_\mathrm{p}^\mathrm{w} = \boldsymbol{b}_\mathrm{schur}$ to guarantee that the effect of the prior node is equivalent to the exact marginalization. We solve the resulting system of equations by inserting $\boldsymbol{H}_\mathrm{p}^\mathrm{w}$ into $\boldsymbol{b}_\mathrm{p}^\mathrm{w}$, which leads to

$$\boldsymbol{b}_\mathrm{p}^\mathrm{w} = \boldsymbol{H}_\mathrm{p}^\mathrm{w}(\boldsymbol{x}_1 - \boldsymbol{x}_\mathrm{p}^\mathrm{w}) \tag{5.16}$$

$$\Leftrightarrow \boldsymbol{b}_\mathrm{schur} = \boldsymbol{H}_\mathrm{schur}(\boldsymbol{x}_1 - \boldsymbol{x}_\mathrm{p}^\mathrm{w}) \tag{5.17}$$

$$\Leftrightarrow \boldsymbol{x}_\mathrm{p}^\mathrm{w} = \boldsymbol{x}_1 - \boldsymbol{H}_\mathrm{schur}^{-1}\boldsymbol{b}_\mathrm{schur}. \tag{5.18}$$

As this allows us to calculate $\boldsymbol{x}_\mathrm{p}^\mathrm{w}$ by using (5.7) and (5.9), we can now compute $\boldsymbol{\Lambda}_\mathrm{p}^\mathrm{w}$:

$$\boldsymbol{\Lambda}_\mathrm{p}^\mathrm{w} = (\breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x})^\top)^{-1}\boldsymbol{H}_\mathrm{schur}\breve{\boldsymbol{J}}_\mathrm{p}(\boldsymbol{x})^{-1}. \tag{5.19}$$

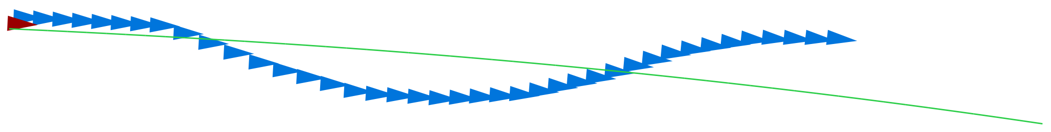
These analytic closed-form expressions for $\boldsymbol{x}_\mathrm{p}^\mathrm{w}$ and $\boldsymbol{\Lambda}_\mathrm{p}^\mathrm{w}$ allow us to position the prior node $\mathrm{x}_\mathrm{p}^\mathrm{w}$ in such a way that the resulting pose estimates for the rest of the graph are identical to the Schur complement marginalization. The mean in combination with the uncertainty estimate provide us with the insight how the marginalization affects the graph. Figure 5.9 illustrates the effect of marginalization and the influence of the prior node for a sliding window estimation of a short trajectory.

As motivated above, marginalization by using a prior node is making the graph construction aware of the marginalization effect, allows us to understand and manipulate the prior information, is efficiently computable, and is thus preferable over marginalization by performing the Schur complement on the system matrix.

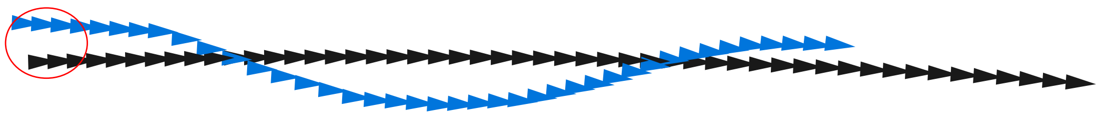
5.4. Timing behavior

After detailing in Section 5.2 how and for which timestamps we construct hidden and observed nodes, we turn our attention to the questions how our system handles the time behavior of input sources and how we design the time behavior of the pose fusion. In this thesis we define time behavior as the latency, frequency, and availability of estimates.

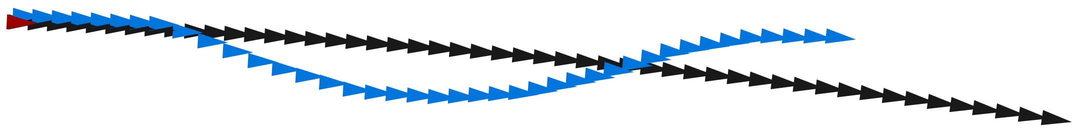
Integrating input sources with unknown time behavior is difficult as we deal with multi-rate sources, nonconstant input frequencies, out-of-sequence estimates, and time-



(a) Unoptimized pose information in the current sliding window from a global pose source (blue), an odometry source (green), and a prior node from marginalization (red). The global pose source drifts and deviates from the odometry source. This is accompanied with a serious reduction of its certainty (not visualized).



(b) Optimized trajectory (black) without a prior node. The weak global pose information is not capable to deform the trajectory. However, the odometry trajectory is fitted to the global poses, leading to a rotated trajectory. This deviation is best seen at the start of the trajectory (red circle).



(c) Optimized trajectory with a prior node. The strong prior node helps to keep the trajectory on track.

Figure 5.9.: The effect of marginalization on a real-world example trajectory. A global pose source (blue) and an odometry source (green) with a potential prior node (red) form the current sliding window optimization problem, as depicted in (a). The result of the optimization of the trajectory without using a prior node is shown in (b), while (c) shows the effect of adding a prior node. The solution with the prior node is less dependent on the erroneous global pose information.

varying latencies. Our approach consists in buffering and preprocessing all incoming data. This does not introduce any delay as we do not need the measurements before the next graph construction phase. The preprocessing detects missing pose estimates by estimating the recent input frequency of each source and comparing the number of estimates we should have received to the number of estimates we actually received. Sorting the data by time enables the integration of out-of-sequence data.

Instead of being data-triggered, the output of the PGF is time-triggered. Its output frequency f is decoupled from the temporal resolution $\varDelta t$. Every $\frac{1}{f}$ the cycle of graph construction and optimization is triggered. The time spent for constructing and opti-

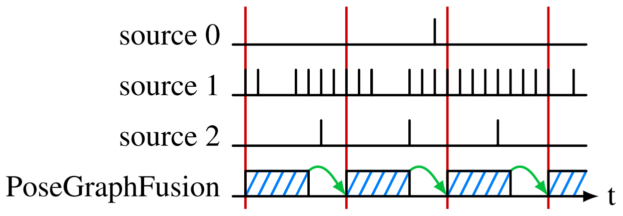


Figure 5.10.: Time behavior of the input sources 0 to 2 and the PGF. The input sources show occasional activity (source 0), data dropouts (source 1), and different data rates. The PGF is able to handle these characteristics and incorporates all input data by first buffering it. At the start of each cycle (directly after the red vertical line) the graph is constructed and optimized. The time necessary for that is the computation time c_t (teal hatched). After the optimization, the estimated pose is propagated to the current point in time.

mizing the graph is summed up as the computation time c_t. The most recent state is estimated with a sliding window pose graph over the current set of measurements. It is subsequently propagated into the current time step with the help of odometry information, as depicted in Figure 5.10. The propagation ensures a low latency of the pose estimate. We define the latency to be the difference between the time when the pose has been computed and the time for which it is valid.

5.5. Resource-adaptive state estimation

In Section 5.2 we have shown that the PGF has a linear runtime complexity in the number of hidden nodes, and we have detailed its time behavior in Section 5.4. We combine this knowledge to enable the algorithm to adapt its parameters at runtime such that it does not use more computational resources than available. This is called a *resource-adaptive* behavior (Thrun et al., 2005). Such an approach becomes especially crucial and valuable when deploying the software in multiple vehicles with different hardware environments such that one parameter set does not fit all of them. We can of course parametrize the computational requirements of the core estimator by changing the length of its sliding window but this is a manual and cumbersome approach. The second motivation for the automated approach is that our sensor fusion software typically shares its hardware resources with other software. This generally means that the available Central Processing Unit (CPU) time varies depending on the workload of the

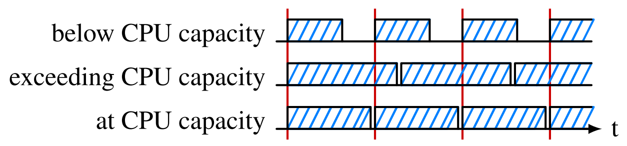


Figure 5.11.: The parametrization of the core estimator can lead to too little, to excessive, and to adequate CPU usage depending on the current CPU load. The red lines mark the beginning of a computation cycle. The computational time c is visualized in dashed blue.

other software. In this case, the parameter set that we might have configured at the start is not reasonable over the entire course of operation.

These resource constraints translate to the challenge that it takes different amounts of time to solve the same problem on different hardware or when the available CPU time fluctuates over time. On the one hand, this potentially results in $c_t > \frac{1}{f}$, i.e., the computation time is greater than the available time in one cycle, which causes a decreased and fluctuating output frequency. On the other hand, the estimation quality is suboptimal if $c_t \ll \frac{1}{f}$ as more hidden nodes could have been considered in the optimization process in the remaining cycle time. Ideally, $c_t = \frac{1}{f}$ for all time steps t. In practice, $\Delta c_t = \frac{1}{f} - c_t \neq 0$ where Δc_t is the remaining cycle time (note that Δc_t can be negative). Figure 5.11 illustrates these different cases of the computational time being too small, too high, and adequate.

The goal is to obtain on average $\Delta c_t \approx 0$ by letting the core estimator adapt the number of hidden nodes M_t in its optimization problem. This results in automatic adaptation to new hardware environments. Moreover, it also adjusts to temporary available or denied CPU resources at runtime.

Our solution consists in using concepts from linear control theory as our system has linear runtime complexity in the number of hidden nodes. We create a feedback loop by measuring the computation time c_t for the current time step with a known number of hidden nodes M_t and subsequently deriving a new number M_{t+1} such that $\Delta c_{t+1} \approx 0$. For this task we employ a proportional-integral-derivative (PID) controller which controls M_{t+1} according to

$$M_{t+1} = M_t + K_\mathrm{p} \Delta c_t + K_\mathrm{i} \sum_{i=1}^{t} \frac{\Delta c_i}{f} + K_\mathrm{d} (\Delta c_t - \Delta c_{t-1}) f, \tag{5.20}$$

where K_p, K_i, and K_d are proportional, integral, and derivative gain constants. These are empirically determined and left constant during the runtime. Note that the sum $\sum_{i=1}^{t} \frac{\Delta c_i}{f}$ does not have to be recomputed from scratch in every time step. Instead, at time step t it is sufficient to add the term $\frac{\Delta c_t}{f}$ to the sum $\sum_{i=1}^{t-1} \frac{\Delta c_i}{f}$, which is known from the previous time step.

Our resource-adaptive state estimation is similar to anytime algorithms[3] in that the quality of results improves gradually as available computation time increases and that we can control computation time. However, other than anytime algorithms we have to control computation beforehand and cannot interrupt an optimization step. The control of the number of hidden nodes leads to a better usage of computational resources and thus to a well-defined time behavior.

5.6. Assessing the uncertainty of the fused estimate

Modeling uncertainty is a key element of probabilistic robotics and obtaining an estimate of the uncertainty of the fused pose estimate is vital for many applications. These include sophisticated motion planners, data association in SLAM, computing the next best view, and other active decisions. We described in the previous sections the construction of a sliding window chain pose graph. The optimization assigns to the hidden nodes the global poses which best satisfy all constraints. In this section we show that we can efficiently recover the uncertainty of the optimized hidden nodes. For this we have already shown in Section 3.4.5 that $\boldsymbol{H}$ is the information matrix of the state given all constraints. Here we will present an algorithm to compute the covariance of all state variables given the sparse Cholesky decomposition of $\boldsymbol{H}$.

Solving the NLLSQ problem of the core estimator with a sparse solver typically involves computing the Cholesky factorization $\boldsymbol{H} = \boldsymbol{R}^\top \boldsymbol{R}$ where $\boldsymbol{R}$ is an upper triangular matrix with entries r_{ij}. Following the derivation of Kaess and Dellaert (2009), the un-

[3]In Computer Science, *anytime algorithms* are algorithms *"[...] that return some answer for any allocation of computation time, and are expected to return better answers when given more time."* (Boddy and Dean, 1989).

certainty matrix of the hidden nodes $\boldsymbol{H}^{-1}$ with entries $\boldsymbol{H}_{ij}^{-1}$ is obtained by the formula

$$\boldsymbol{H}_{ii}^{-1} = \frac{1}{r_{ii}} \left[\frac{1}{r_{ii}} - \sum_{\substack{k=i+1 \\ r_{ik} \neq 0}}^{n} r_{ik} \boldsymbol{H}_{ik}^{-1} \right], \tag{5.21}$$

$$\boldsymbol{H}_{ij}^{-1} = \frac{1}{r_{ii}} \left[- \sum_{\substack{k=i+1 \\ r_{ik} \neq 0}}^{j} r_{ik} \boldsymbol{H}_{kj}^{-1} - \sum_{\substack{k=j+1 \\ r_{ik} \neq 0}}^{n} r_{ik} \boldsymbol{H}_{jk}^{-1} \right], \tag{5.22}$$

where the other half of the matrix can be obtained by exploiting the symmetry of $\boldsymbol{H}^{-1}$. Assuming that we compute both the upper and lower triangular part of $\boldsymbol{H}^{-1}$ and exploiting the symmetry $\boldsymbol{H}_{jk}^{-1} = \boldsymbol{H}_{kj}^{-1}$, we can develop this to

$$\boldsymbol{H}_{ii}^{-1} = \frac{1}{r_{ii}} \left[\frac{1}{r_{ii}} - \sum_{\substack{k=i+1 \\ r_{ik} \neq 0}}^{n} r_{ik} \boldsymbol{H}_{ik}^{-1} \right], \tag{5.23}$$

$$\boldsymbol{H}_{ij}^{-1} = \frac{1}{r_{ii}} \left[- \sum_{\substack{k=i+1 \\ r_{ik} \neq 0}}^{n} r_{ik} \boldsymbol{H}_{kj}^{-1} \right]. \tag{5.24}$$

This formula yields an $\mathcal{O}(n)$ time complexity (with n being the number of state variables) because it operates on the nonzero entries of the sparse band matrix $\boldsymbol{R}$. It becomes constant time for sliding window pose graphs as the number of state variables is upper bounded by a constant value. Furthermore, we abort the computation of $\boldsymbol{H}^{-1}$ once we obtain the covariance matrix for the elements of interest. As we are only interested in the estimated uncertainty of the last hidden node, only a single and comparably trivial calculation has to be performed:

$$\boldsymbol{H}_{nn}^{-1} = \frac{1}{r_{nn}} \frac{1}{r_{nn}}. \tag{5.25}$$

We can therefore compute the uncertainty of our fused pose estimates and even more, do so efficiently.

6. Preprocessing sublayer

This chapter presents the preprocessing sublayer and its modules. These modules are designed such that they are as broadly applicable as possible. They are of course tailored towards the core estimator but can also be extracted and applied to similar contexts. These include other graph- or filtering-based approaches.

The preprocessing sublayer consists of the four modules bias estimation, outlier handling, cross-correlated and autocorrelated noise treatment, as shown in Figure 6.1. These are presented in the following sections.

6.1. Bias estimation

The core estimator assumes that all input pose estimates are unbiased. However, a bias in the estimates of a global pose source is common (Jo et al., 2013; Laneurit et al., 2005; Tao et al., 2013). It results in a mean error unequal to zero when compared to the true trajectory. In this section we deal with quasi-stationary and with systematic biases. Quasi-stationary biases are constant for a limited series of estimates while systematic

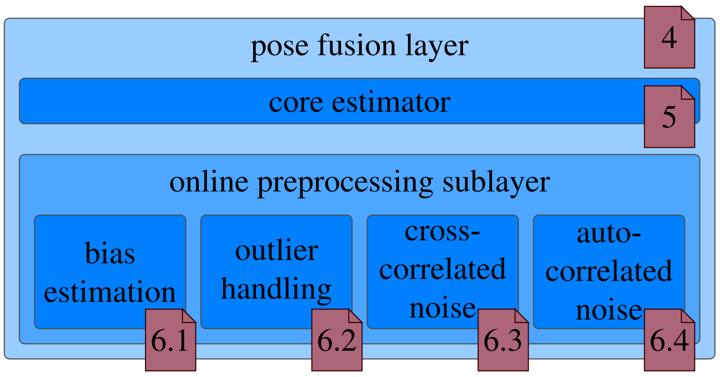


Figure 6.1.: Modules in the preprocessing sublayer.

biases are constant for all estimates. Sometimes, quasi-stationary biases are also called time-varying biases. Figure 6.2 illustrates the effects of unbiased and time-varying biased pose estimates. Biases of this kind result in decreased performance of the core estimator: the mean error of the fused estimate will generally be unequal to zero and the estimator potentially diverges. It is therefore important to reduce biases as much as possible.

Within our fusion architecture, it seems at first sight appealing to delegate the task of bias estimation and removal to the pose sources. After all, they know best about their sensors, internal working principles, and possible sources of biases. However, they sometimes simply lack the necessary information to estimate a bias. We assume that they clean their estimates from biases that they are able to observe, compute, or estimate. Subsequently, our preprocessing module takes care of eliminating the remaining part of biases as much as possible. It has the advantage over the pose sources that it has access to pose estimates from other sources. This additional knowledge helps to reduce biases by means of comparison.

We propose an effective technique to estimate quasi-stationary biases at runtime. It is also applicable to stationary biases as they can be regarded as a subset of quasi-stationary biases. This technique enables pose sources to better satisfy the noise assumptions by reducing the mean error of their estimates, thus increasing their consistency and accuracy.

Quasi-stationary biases vary over time. The duration in which such a bias stays roughly constant depends on the specific pose source and its sensor, its field of application, and environmental conditions. A prominent example of a class of global pose sources which commonly suffers from time-varying biases are pose estimates based on GPS data. Clock errors and multipath effects are common sources for this bias (Jo et al., 2013). The bias might for example stay roughly constant as long as the GPS receiver sees the same constellation of satellites or suffers from the same multipath effects. Other, rather sensor-independent, sources of systematic error include calibration and time synchronization issues. Also, offsets in the map lead to systematic biases for map-based global pose sources. The key contribution of this chapter is an online bias estimation technique for quasi-stationary and systematic biases of global pose sources that is fast, effective, and straightforward to implement.

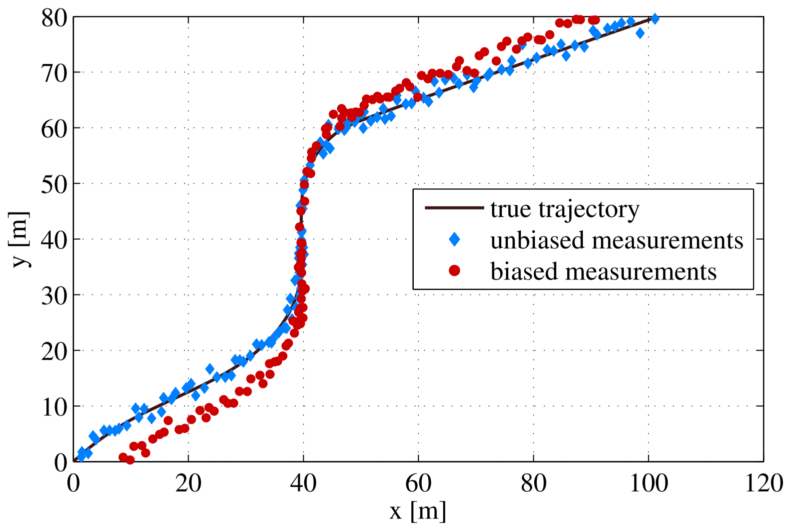


Figure 6.2.: Illustration of the effects of a time-varying bias. The true trajectory (black line) is measured by an unbiased (blue diamonds) and a biased (red dots) pose source.

6.1.1. Comparison of unbiased and biased pose estimates in a sliding window

Our goal is to derive a method that allows us to estimate the bias of a global pose source and subsequently to remove it from the pose estimates. Our generic pose fusion approach does not assume any knowledge about the implementation details of the input sources, which makes it challenging to eliminate the unknown source of biases. We therefore propose a generic bias estimation method. To this end, we analyze the global pose estimates $\boldsymbol{z}^b$ with information matrices $\boldsymbol{\Lambda}^b$ from a biased source and compare them to a source which is noisy but produces unbiased estimates[1] $\boldsymbol{z}^u$ with information matrices $\boldsymbol{\Lambda}^u$. More formally, we model the noise as

$$\boldsymbol{z}_i^u = \boldsymbol{p}_i + \boldsymbol{v}_i^u \tag{6.1}$$

$$\boldsymbol{z}_i^b = \boldsymbol{p}_i + \boldsymbol{v}_i^b + \boldsymbol{c}_i^b, \tag{6.2}$$

[1]The superscript b stands for *biased*, and u for *unbiased*.

where $\boldsymbol{p}_i$ is the true pose for the i-th pose estimate, $\boldsymbol{v}^u$ and $\boldsymbol{v}^b$ are white noise such that $\boldsymbol{v}^u \sim \mathcal{N}(\mathbf{0}, (\boldsymbol{\Lambda}^u)^{-1})$ and $\boldsymbol{v}^b \sim \mathcal{N}(\mathbf{0}, (\boldsymbol{\Lambda}^b)^{-1})$, and $\boldsymbol{c}_i^b$ denotes the bias. Equation (6.2) is a common noise model and we will show how to gain knowledge about $\boldsymbol{c}_i^b$ by observing the corresponding unbiased pose source governed by (6.1).

The key idea of the bias estimation is that the difference of the mean estimates of a biased source compared to an unbiased source is roughly equal to the respective bias. Estimating the bias in a sliding window of length s, such that the estimated bias $\hat{\boldsymbol{c}}_i^b(s)$ for the i-th pose estimate is a function of s, allows it to adapt to time-varying biases. The estimate of the bias $\hat{\boldsymbol{c}}_i^b(s)$ also depends on the unbiased pose source u, but we omit the additional index for the sake of notational clarity. We refine this method by computing the mean errors as weighted averages such that pose estimates with higher variances account for less impact. In total, we define the estimated bias $\hat{\boldsymbol{c}}_i^b(s)$ accordingly as

$$\hat{\boldsymbol{c}}_i^b(s) = \frac{1}{s}\frac{1}{\sum_{j=i-s}^{i}\boldsymbol{\Lambda}_j^u}\sum_{j=i-s}^{i}\boldsymbol{\Lambda}_j^u\boldsymbol{z}_j^b - \frac{1}{s}\frac{1}{\sum_{j=i-s}^{i}\boldsymbol{\Lambda}_j^u}\sum_{j=i-s}^{i}\boldsymbol{\Lambda}_j^u\boldsymbol{z}_j^u \tag{6.3}$$

$$= \frac{1}{s}\frac{1}{\sum_{j=i-s}^{i}\boldsymbol{\Lambda}_j^u}\sum_{j=i-s}^{i}\boldsymbol{\Lambda}_j^u(\boldsymbol{z}_j^b - \boldsymbol{z}_j^u). \tag{6.4}$$

Figure 6.3 illustrates the application of this method. Having computed $\hat{\boldsymbol{c}}_i^b(s)$, we can subsequently subtract it from the biased pose estimates $\boldsymbol{z}_i^b$ to eliminate their impact. This technique is straightforward and effective as we show in the evaluation section.

A key issue is the determination of the size s of the sliding window. On the one hand, the estimated bias adapts too slowly to the true time-varying bias if the window size is too large. On the other hand, we violate the assumption that the unbiased pose source has a zero-mean error over the sliding window if the sliding window size is too small. A sliding window size that is too small additionally leads to a significant correlation[2] of the error terms of the biased and unbiased source.

We formulate the search for the optimal sliding window size s^* as an offline optimization over a dataset with L data points per pose source. This dataset contains unbiased estimates $\boldsymbol{z}^u$, biased estimates $\boldsymbol{z}^b$, and ground truth poses $\boldsymbol{p}$. The optimal sliding win-

[2]In Section 3.5.1 we provide an insight on the order of the error of the uncertainty estimate given a certain cross-correlation. This allows us to gauge what cross-correlation is tolerable.

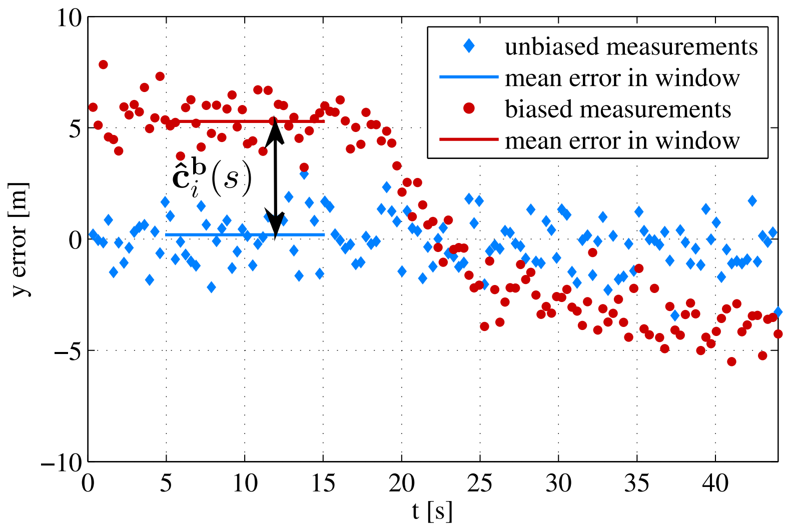


Figure 6.3.: Measurement error over time of the unbiased (blue diamonds) and biased (red dots) pose sources. The difference of their mean errors in a sliding window of $10\,\mathrm{s}$ leads to the computation of $\hat{\boldsymbol{c}}_i^b(s) \approx 5\,\mathrm{m}$.

dow size s^* is given by

$$s^* = \arg\min_s \frac{1}{L} \sum_{i=1}^{L} \left\| (\boldsymbol{z}_i^b - \hat{\boldsymbol{c}}_i^b(s)) - \boldsymbol{p}_i \right\|, \tag{6.5}$$

$$\text{s.t.} \qquad \rho(\boldsymbol{z}^b - \boldsymbol{p}, \boldsymbol{z}^u - \boldsymbol{p}) \leq \rho_{\text{tol}}, \tag{6.6}$$

$$\tau(\boldsymbol{z}^u, \boldsymbol{p}, s) \leq \tau_{\text{tol}}. \tag{6.7}$$

The first condition given by (6.6) is true if the errors of the bias-corrected source and the unbiased source are correlated less than a threshold ρ_{tol}. For this $\rho(\cdot,\cdot)$ is an auxiliary function which computes the correlation between two input vectors. The second condition given by (6.7) is true if the sliding window is large enough to justify the assumption of a zero-mean error of the unbiased source. $\tau(\cdot,\cdot,\cdot)$ is an auxiliary function such that 95% of all terms $\frac{1}{s}\sum_{j=i-s}^{i} \left\| \boldsymbol{\Lambda}_j^u \boldsymbol{z}_j^u - \boldsymbol{p}_j \right\|$ for $i = 1, \ldots, L$ are less or equal than its function value. The usage of the 95% percentile robustifies the function against outliers. As this is an offline optimization without timing requirements, we content ourselves to solve (6.5) with an exhaustive search over all sliding window sizes of interest. Nevertheless, this is typically done in the order of seconds.

This approach can easily be extended to incorporate multiple unbiased pose sources by estimating $\hat{c}^b(s)$ for each of these sources. In general, we assume that the majority of sources are unbiased. We can easily determine whether a source is biased by observing all pose sources over a certain period of time and performing a simple version of random sample consensus.

6.2. Map-based outlier handling

Outliers can have serious impact on classical estimation methods. They tend to skew the result towards their values. For generic pose fusion, the risk is to distort or displace the estimated trajectory. Therefore, it is important to handle outliers appropriately.

Generally speaking, an outlier is an observation that is far away from all other observations. In our case, this means one of two things: either a single pose estimate is far away from all other pose estimates or a single pose source continuously produces estimates far away from all other pose sources. In the latter case, one could either classify the entire pose source as outlier or all of its data. We design our outlier handling to be able to deal with both cases.

When performing generic pose fusion it is difficult to tell when a specific pose source produces unreliable pose estimates because we lack knowledge of the underlying sensors and algorithms. Therefore, we present a method for examining global pose estimates that is sensor- and algorithm-independent. It solely analyzes the pose estimates of any given pose source.

The core estimator is based on solving a weighted optimization problem. These weights are chosen as the information matrices of the pose estimates. They are of crucial importance as they adjust the amount of information that a single pose estimate contributes to the fused estimate. We present a method for scaling the covariance matrix of an input pose estimate given prior knowledge in the form of a map. We adapt the covariance matrix if the pose estimate's location on a map seems unreasonable. That is, we leverage the knowledge that a car is usually driving on the road or close to it to judge the quality of a pose estimate. The effect of this method is that we can detect and smoothly handle outliers simply by scaling their covariance matrices.

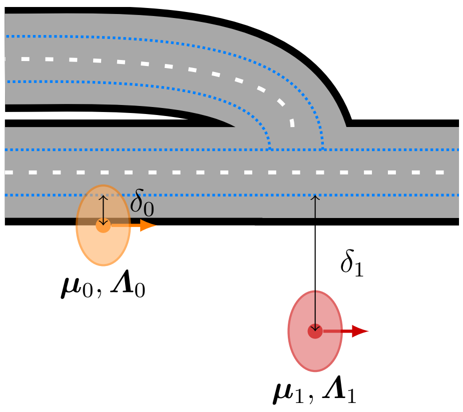
The key idea is that if we know where the vehicle should be driving right now then we can infer which pose estimates seem unlikely and treat them more carefully. The question is how do we know where the vehicle should be driving. It turns out that we can gain information about this in multiple ways. First, in an automated vehicle we might be able to leverage the planned trajectory of the vehicle. Secondly, in automated platooning or similar applications the vehicle is often given a trajectory that it should follow. Thirdly, it is usually safe to assume that the vehicle is driving on a road. In the following we focus on the latter possibility. In the evaluations in Section 7.4.2 we show that this assumption is broadly satisfied. We employ maps with a precise road geometry as presented in Section 3.3.2. Specifically, we exploit the concept of center lines. They give us the geometric middle of each lane, which is close to where most vehicles are driving.

With this information in hand, our approach compares each global pose estimate to the center lines of a DLM and scales its information matrix accordingly. Our intuition is that a pose estimate is more likely corrupted as that the vehicle is driving far off the road. In this sense, we exploit the map information to judge the global pose estimates.

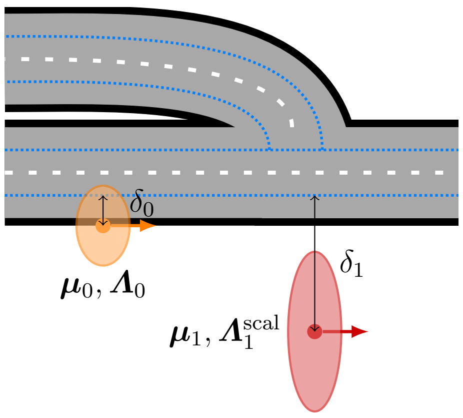
If we have detected a pose estimate as an outlier, we prefer to scale its information matrix to decrease its influence in the fusion over completely rejecting the outlier. This has two reasons. First, the decision to reject a pose estimate requires a hard criterion of what we consider an outlier. It is tough to design such a criterion with a low probability of false alarm. In contrast, our method of scaling provides a smooth separation between inliers and outliers. How much this information comes into play in the core estimator depends on the uncertainties of all other pose estimates. We see this approach therefore in the same line of thought as robust cost functions (cf. Section 3.4.3) which downscale the error terms instead of rejecting too high values. A similar effect is achieved when upscaling the information matrices.

Secondly, we note that our approach is a heuristic that judges independently of knowledge of the sensors' and algorithms' working principles and can falsely detect outliers. Imagine the case of an emergency maneuver where the vehicle is forced to leave the road and come to a stop on the side of it. In this case, all of the pose sources will hint at this fact. If we were to reject all pose estimates because they are sufficiently far away from the road, then we have trouble estimating the vehicle's pose. In contrast, we propose to scale the uncertainties of this information such that we can compute the fused pose as being off the road and as being uncertain, which is a favorable behavior.

Our key idea consists of choosing an appropriate scaling function, computing the


δ_0
$\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0$
δ_1
$\boldsymbol{\mu}_1, \boldsymbol{\Lambda}_1$

(a) Two pose estimates are given of which the orange one is close and the red one is far away from the road. Initially, they both have the same covariance matrix.


δ_0
$\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0$
δ_1
$\boldsymbol{\mu}_1, \boldsymbol{\Lambda}_1^{\text{scal}}$

(b) After applying our scaling, the covariance matrix of the red pose estimate is scaled up. This reflects our semantic knowledge that we generally expect vehicles to drive on the road.

Figure 6.4.: Key idea of the map-based outlier handling: inappropriately small covariance matrices of pose estimates far away from the center line are scaled up.

distance of a pose estimate to the center line that it likely belongs to, and scaling the estimate's information matrix accordingly. Figure 6.4 illustrates this idea. In the following we detail the steps behind it.

6.2.1. Scaling the information matrix of a pose estimate

First of all, it is important to compute the distance of the pose estimate to the corresponding center line of the road. As detailed in Section 3.3.2, each segment of a DLM has one or more lanes. Each lane has a center line which is modeled as a polyline. Thus, for a given pose estimate we examine all center lines in its close vicinity to find the one that the pose estimate most likely belongs to. Once we identify this center line we compute the minimal lateral, longitudinal, and heading distance to it by orthogonal projection. Figure 6.5 illustrates a pose estimate (blue), the corresponding center line (green), and the orthogonal projection on it. Note that we take heading difference as the difference between the pose estimate's heading and the direction of the center line (by also taking into account the known driving direction). Usually, this comparison with a map allows

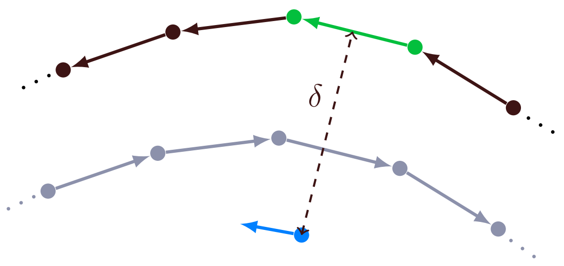


Figure 6.5.: Computing the distance δ (dashed) of a pose estimate (blue) to the center line. The center line is depicted for both driving directions, indicated by arrow heads. The green part of the center line is the lane piece that has the shortest distance to the pose estimate. While lane pieces of the gray center line are closer to the pose estimate, the driving direction does not match and therefore the black center line is chosen.

us to get good information about the lateral deviation of the pose estimate to the center line, but the longitudinal and heading distances are less precise (see the discussion of Wijesoma et al. (2006) for the observability of path constrained vehicle localization). We consider this when designing our scaling functions by choosing more conservative parameters for the longitudinal and heading distances. Overall, the lateral localization error is the most interesting error component and therefore a precise determination of the lateral distance is of foremost interest.

Consider the i-th pose estimate with its information matrix $\boldsymbol{\Lambda}_i$. Let the minimal longitudinal, lateral, and heading distances of the pose estimate to the center line be δ_x, δ_y, and δ_θ. For now we assume that a scaling matrix $\boldsymbol{S}$ exists such that its Cholesky decomposition is

$$\boldsymbol{S} = \begin{bmatrix} s_x(\delta_x) & 0 & 0 \\ 0 & s_y(\delta_y) & 0 \\ 0 & 0 & s_\theta(\delta_\theta) \end{bmatrix} = \boldsymbol{L}_S \boldsymbol{L}_S^\top, \tag{6.8}$$

where s_x, s_y, and s_θ are scaling functions that depend upon the respective distances. We propose to scale $\boldsymbol{\Lambda}_i$ such that

$$\boldsymbol{\Lambda}_i^{\text{scal}} = \boldsymbol{L}_S \boldsymbol{\Lambda}_i \boldsymbol{L}_S^\top. \tag{6.9}$$

This slightly complicated notation allows us to express that all entries of $\boldsymbol{\Lambda}_i$ are scaled.

The remaining question is how to define s_x, s_y, and s_θ.

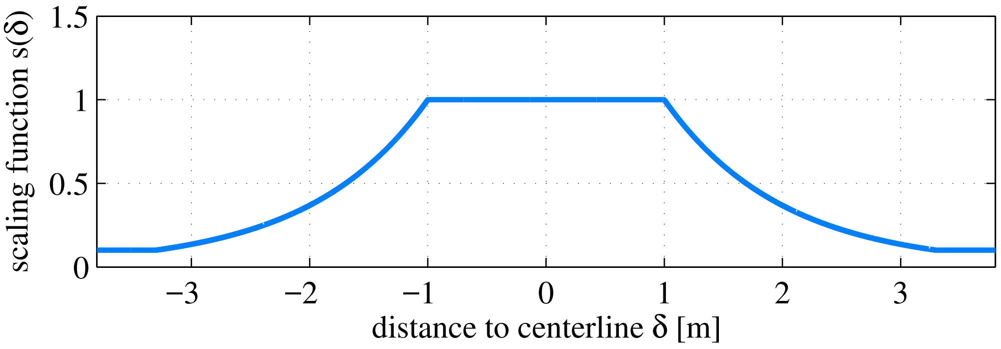


Figure 6.6.: Scaling function as defined by (6.10) for different distances to the center line. The parameters are $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $s_{\text{min}} = 0.1$, and $s_{\text{max}} = 1.0$.

6.2.2. Scaling functions

Different choices are possible for the scaling functions. Similar to robust cost functions, the optimal choice depends upon the concrete situation. The idea of robust cost functions is to limit the influence of outliers by reducing their weight in the optimization process. To this end, they scale large error terms less than the default quadratical scaling. See Section 3.4.3 for more information. Our scaling functions are similar in that they reduce the influence of potential outliers by increasing their covariance.

We present one parametrized scaling function that works well in our context. Dropping the indices on the scaling functions s and on the distances δ for notational clarity we define

$$s(\delta) = \begin{cases} s_{\text{min}} & \exp\left(\frac{\lambda_1 - |\delta|}{\lambda_2}\right) \leq s_{\text{min}} \\ s_{\text{max}} & \exp\left(\frac{\lambda_1 - |\delta|}{\lambda_2}\right) \geq s_{\text{max}} \\ \exp\left(\frac{\lambda_1 - |\delta|}{\lambda_2}\right) & \text{else,} \end{cases} \tag{6.10}$$

with parameters λ_1, λ_2, s_{min}, and s_{max}. This seemingly complex definition of s is quickly untangled when plotting the corresponding function. Figure 6.6 shows this function for a certain parameter set. We see that s_{min} defines a minimum value below which the scaling does not drop. If we wanted to reject extremely improbable pose estimates, then we could set this value to zero. However, we will generally prefer to set this to a minimum value above zero to respect the probability of false alarm. Similarly, s_{max} defines the maximum value of the scaling. It will often make sense to limit this to 1 as the scaling would otherwise imply that the pose estimate contains more information than it claims itself. The parameters λ_1 and λ_2 define the slope of the scaling. It allows us to use steeper slopes for pose estimates with little uncertainty. Also, their definition leads to a plateau around $|\delta| \approx 0$. The reasoning behind this is that we do not want to penalize pose estimates near the center line as it is highly likely that the vehicle is driving *close to* and not *exactly on* the center line. The scaling comes into effect for pose estimates that are sufficiently far away from the center line where "sufficiently" is defined by the width of the plateau. This width is determined empirically by examining recorded data to analyze usual distances of the vehicle to the center line. We show a corresponding experiment in Section 7.4.2. The set of all four parameters $(s_{\min}, s_{\max}, \lambda_1, \lambda_2)$ is found empirically. For our use case, the exact form of the scaling function and its parameters did not turn out to be of significant importance. Therefore, we restrict ourselves to a single scaling function instead of comparing several ones.

6.3. Cross-correlated errors between pose sources

Multi-sensor fusion based on NLLSQ optimization commonly assumes uncorrelated noise between measurements. However, the noise of different pose sources can in general be correlated. Ignoring correlated noise leads to overconfident pose estimates, corrupt uncertainty estimates, and potentially estimator divergence. Noise is correlated between pose sources when, for example, the same sensor or map data is being used in different algorithms or when the same algorithm runs on two physically different sensors (e.g., two GPS receivers). Two of the major sources are *common process noise* and *common prior information* (Reinhardt et al., 2012). Both lead to common noise that potentially influences all affected sources. Ideally, we would be able to eliminate the source of the common noise, but as we perform generic pose fusion we lack the insight what this common source is. Therefore, we try to minimize its impact in the fusion process by using CI to produce conservative estimates.

Figure 6.7 motivates why naive fusion is suboptimal as its result is overconfident. The true correlation between $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ is unknown and randomly drawn positive correlations (displayed in gray) show that the two CI methods (trace minimization with $\omega = 0.69$ and determinant minimization with $\omega = 0.29$) provide conservative bounds.

In this section we detail our approach of treating correlated noise between pose sources with the help of a CI framework. We refer to the presentation of CI in Sec-

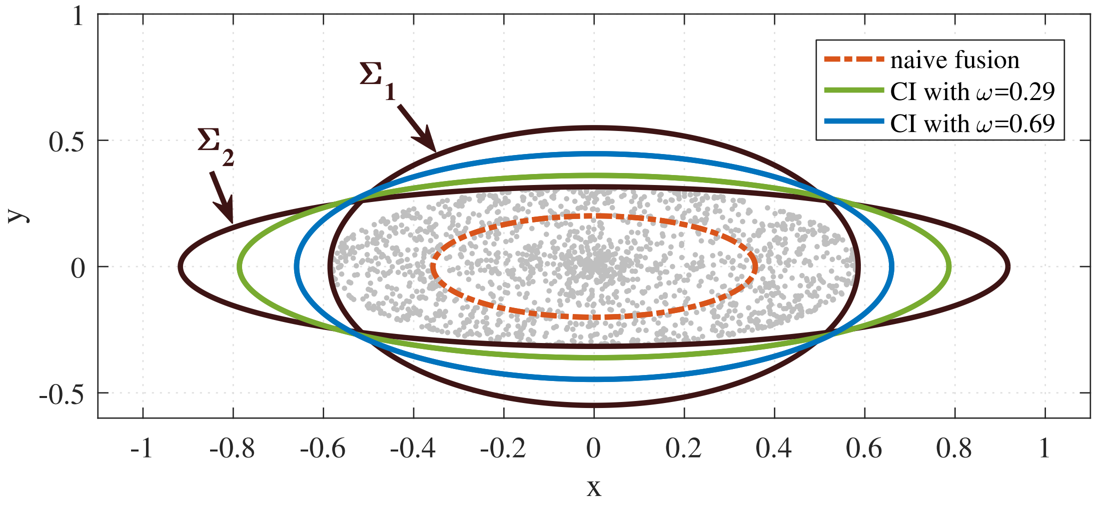


Figure 6.7.: $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ are two example covariance matrices with unknown correlation that we want to fuse. CI covariance ellipses for trace ($\omega = 0.69$) and determinant ($\omega = 0.29$) minimization are shown in blue and green. True covariance values for randomly chosen correlations are displayed as gray dots. The naive fusion (dashed, red) is potentially overconfident. Note that gray dots within the ellipse of the naive fusion correspond to negative correlation coefficients, which have been included for the sake of completeness.

tion 3.5 for this. Generally, if we have two pose sources with cross-correlated noise, we apply the closed-form solutions given in Section 3.5.4 to produce a single conservative estimate. We extend these formulas so that we can apply them to more general matrices. This includes two diagonal covariance matrices that share at least one common entry, for example.

6.3.1. Extension of the closed-form solutions of Covariance Intersection

The closed-form expressions for trace and determinant minimization in Section 3.5.4 are derived independent of the properties of the covariance matrices to which they are applied. We give a simple example in which they cannot be applied, highlight the underlying problem, and show how we can reduce the dimensionality of the optimization problem for both trace and determinant minimization so that we can again apply the analytic solutions. Here we pickup the notation from Section 3.5.4. We recall that d_i are the eigenvalues of the matrix $\boldsymbol{\Sigma}_2'$, which we obtained as a result of the joint diagonalization of the two covariance matrices of interest.

The issue is that we can only set $\tilde{d}_i = \frac{\bar{d}_i}{1-\bar{d}_i}$ if $\bar{d}_i \neq 1$. This, however, is not always the case as the following example shows. Without loss of generality let

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} \boldsymbol{\Sigma}_1^* & \mathbf{0} \\ \mathbf{0} & \sigma^2 \end{bmatrix}, \qquad \boldsymbol{\Sigma}_2 = \begin{bmatrix} \boldsymbol{\Sigma}_2^* & \mathbf{0} \\ \mathbf{0} & \sigma^2 \end{bmatrix} \tag{6.11}$$

with $\sigma^2 > 0$. We then verify that $\bar{d}_N = 1$ (with N being the problem dimensionality). This is the case whenever at least one element of $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ is identical. For notational simplicity we assume in the following that this is the last element in $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$. This important class of matrices is of interest to us. We will now investigate how to treat the case $\bar{d}_i = 1$ for both trace and determinant minimization so that we can extend the CI solution to this class of matrices.

First, we are interested in determinant minimization and directly develop (3.84) to

$$\omega^* = \arg\max_{\omega} \prod_{i=1}^{N} \omega + (1-\omega)\bar{d}_i \tag{6.12}$$

$$= \arg\max_{\omega} \prod_{i=1}^{N-1} \omega + (1-\omega)\bar{d}_i. \tag{6.13}$$

Hence, the dimension of the CI problem can be reduced by one by solving the CI problem for $(\boldsymbol{\Sigma}_1^*, \boldsymbol{\Sigma}_2^*)$ instead of $(\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2)$.

Secondly, we turn to trace minimization and develop (3.87) to

$$\omega^* = \arg\max_{\omega} \sum_{i=1}^{N} \frac{a_i}{\omega + (1-\omega)\bar{d}_i} \tag{6.14}$$

$$= \arg\max_{\omega} \sum_{i=1}^{N-1} \frac{a_i}{\omega + (1-\omega)\bar{d}_i} + \frac{a_n}{1} \tag{6.15}$$

$$= \arg\max_{\omega} \sum_{i=1}^{N-1} \frac{a_i}{\omega + (1-\omega)\bar{d}_i}. \tag{6.16}$$

Again we observe that this case leads to the reduction of the dimensionality of the problem. The key insight is that this problem is now well-defined and that we can apply our common set of CI equations to solve it.

This derivation is easily extended to the case that $\bar{d}_i = 1$ for multiple i. This reduces the set of viable candidates for ω^* by the amount of i for which $\bar{d}_i = 1$. In the extreme case that $\bar{d}_i = 1$ for all $i = 1, \ldots, N$ all eigenvalues of $\boldsymbol{E}_2'$ are equal to 1 and therefore the identity

$$\boldsymbol{T}\boldsymbol{\Sigma}_2\boldsymbol{T}^\top = \boldsymbol{E}_2' = \boldsymbol{I} = \boldsymbol{T}\boldsymbol{\Sigma}_1\boldsymbol{T}^\top \tag{6.17}$$

holds which implies $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$ since $\boldsymbol{T}$ is invertible. Therefore, $\omega \in [0,1]$ can be set arbitrarily since $\boldsymbol{\Sigma}^\omega = \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$ holds.

With these derivations we are able to apply CI to any two covariance matrices regardless of their entries. Our contribution consists in showing how to extend the closed-form solutions for CI with both trace and determinant minimization to matrices which share at least one common entry.

6.4. Measurements with autocorrelated errors

The core estimator assumes that the noise of all input pose estimates is AWGN. The assumption of white noise implies that all pose estimates are *independently* drawn from Gaussian distributions. In this section we are interested in those sources for which this assumption does not hold because their error is autocorrelated. If this autocorrelation is of order 1, then we can model the noise of the pose source as AR(1). This has important implications on the estimation procedure. Also, autocorrelated errors are common for prefiltered pose sources or those where the observations themselves are correlated over time. The latter is true for observations of GNSS receivers. These can either be modeled as AR(1) (Zangeneh-Nejad et al., 2015) or, more sophisticated, as autoregressive moving average processes (Luo, 2013). In the context of this work, we are interested in how to preprocess estimates of pose sources whose noise can be modeled as an AR(1) process.

We refer to NLLSQ estimation which does not consider autocorrelated errors as *ordinary NLLSQ*[3]. In contrast, we refer to NLLSQ estimation with autocorrelated errors as AR(1) NLLSQ. We derive this concept in the following. It is applicable to graph-based estimation problems, such as our pose fusion, SLAM, bundle adjustment, or other NLLSQ problems. If we apply ordinary NLLSQ to data with autocorrelated errors, then the estimator is still unbiased but inefficient. Even more, the estimated covariances are substantially biased (Maddala and Lahiri, 1992). Moreover, statistical tests that rely on the estimated covariances are invalid. It is therefore important to correctly model autocorrelated errors.

We derive in Section 6.4.1 the AR(1) NLLSQ scheme. This results in a correct estimation of the uncertainty of the output at the cost of a higher computational demand and the loss of a graph-based representation. With this result in mind, we subsequently show two ways to incorporate this knowledge of the AR(1) NLLSQ into the ordinary NLLSQ estimation process. The first one in Section 6.4.2 derives an understanding of the influence of autocorrelated noise on the graph itself. We show how to use nodes and

[3]Sometimes, it is referred to as *ordinary least squares (OLS)* for both linear and nonlinear problems.

edges to construct the same effect as autocorrelated errors in the resulting optimization problem. Thus, we gain an understanding of how pose estimates with autocorrelated errors can be modeled in a graph purely with the help of graph elements. However, at this point we still suffer from an increased computational demand. To tackle this challenge, we seek to keep the optimization problem as close to an ordinary NLLSQ procedure as possible. To this end, the second method is described in Section 6.4.3 and demonstrates that scaling the information matrices of the pose estimates is exactly equivalent to solving the AR(1) NLLSQ problem in terms of the estimated covariance matrix. It allows us to efficiently solve the problem as the sparsity is exactly preserved.

In total, we therefore first show how autocorrelated errors can be modeled within the optimization problem. Secondly, we derive a graph-based representation of this. Thirdly, we show how to efficiently implement this.

6.4.1. Nonlinear least squares with autocorrelated errors

Our goal is now to derive the normal equations of AR(1) NLLSQ estimation. In Section 3.4.2 we derived the constraint-based formulation of the NLLSQ estimation by summing up over all single constraints. This implicitly assumes that the noise of all constraints is uncorrelated. As we drop this assumption in the following, it is instrumental to prefer a vectorized notation. Making use of our description of the geodetic formulation in Section 3.4.7, we recall that

$$\boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{A} \hat{\boldsymbol{x}} = \boldsymbol{A}^\top \boldsymbol{P} \boldsymbol{l} \tag{6.18}$$

is already in vectorized form. A similar derivation can be made for the constraint-based notation with the goal of writing it in a vectorized form as

$$\breve{\boldsymbol{J}}^\top \boldsymbol{\varSigma}^{-1} \breve{\boldsymbol{J}} \varDelta \boldsymbol{x}^* = -\breve{\boldsymbol{J}}^\top \boldsymbol{\varSigma}^{-1} \breve{\boldsymbol{e}}. \tag{6.19}$$

These are two ways of denoting the same equation. Here we stack all terms $\breve{\boldsymbol{J}}_i$ in one big matrix $\breve{\boldsymbol{J}} = [\dots, \breve{\boldsymbol{J}}_i^\top, \dots]^\top \in \mathbb{R}^{3n \times 3m}$, where n is the number of constraints, m the number of states, and the factor 3 arises due to the dimension of a state variable. Similarly, we stack the error vector into $\breve{\boldsymbol{e}} = [\dots, \breve{\boldsymbol{e}}_i^\top, \dots]^\top \in \mathbb{R}^{3n \times 1}$. We form the matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3n \times 3n}$ equivalently by creating a block-diagonal matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \ddots & & \\ & \boldsymbol{\Sigma}_i & \\ & & \ddots \end{bmatrix}. \tag{6.20}$$

This forms the total stochastic model of our estimation problem. The block-diagonal form is often chosen because its inverse is easily computable and it preserves the sparsity of the problem (given that $\breve{\boldsymbol{J}}$ is sparse). However, when dealing with autocorrelated errors we have to modify $\boldsymbol{\Sigma}$ to reflect the temporal dependency of the noise.

We model the measurements and their errors of a pose source with autocorrelated noise as

$$\boldsymbol{z}_i^{\mathrm{AR}} = \boldsymbol{p}_i + \boldsymbol{\epsilon}_i^{\mathrm{AR}}, \tag{6.21}$$

$$\boldsymbol{\epsilon}_i^{\mathrm{AR}} = \phi \boldsymbol{\epsilon}_{i-1}^{\mathrm{AR}} + \boldsymbol{v}_i, \tag{6.22}$$

where $\boldsymbol{z}_i^{\mathrm{AR}}$ is the i-th measurement of the true pose $\boldsymbol{p}_i$ with the additive error vector $\boldsymbol{\epsilon}_i^{\mathrm{AR}}$. This error vector is described as an AR(1) process with autocorrelation coefficient ϕ and AWGN $\boldsymbol{v}_i \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{v}})$.

In Section 3.6 we highlighted ways to judge whether a given time series can be considered to stem from an AR(1) process. Additionally, we proposed ways to estimate ϕ. Here we assume that we successfully applied this knowledge to verify that we can model the error of the measurements of a given pose source as an AR(1) process with the autocorrelation coefficient ϕ. Let us assume covariance stationarity for this process such that its covariance is identical for all time steps. For the simplicity of the following derivation, we assume equal variances $\boldsymbol{\Sigma}_{\boldsymbol{v}}$ for all measured quantities. The joint covariance matrix $\boldsymbol{\Sigma}^{\mathrm{AR}}$ for all measurements with autocorrelated noise is then

$$\boldsymbol{\Sigma}^{\mathrm{AR}} = \frac{1}{1-\phi^2} \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{v}} & & \\ & \ddots & \\ & & \boldsymbol{\Sigma}_{\boldsymbol{v}} \end{bmatrix} \begin{bmatrix} 1 & \phi\mathbf{1} & \phi^2\mathbf{1} & \cdots & \phi^{n-1}\mathbf{1} \\ \phi\mathbf{1} & \mathbf{1} & \phi\mathbf{1} & & \phi^{n-2}\mathbf{1} \\ \phi^2\mathbf{1} & \phi\mathbf{1} & \mathbf{1} & & \vdots \\ \vdots & & & \ddots & \phi\mathbf{1} \\ \phi^{n-1}\mathbf{1} & \phi^{n-2}\mathbf{1} & \cdots & \phi\mathbf{1} & \mathbf{1} \end{bmatrix}, \tag{6.23}$$

where the latter matrix is the so-called *correlation matrix*. Interestingly, the inverse of the dense matrix $\boldsymbol{\Sigma}^{\mathrm{AR}}$ is sparse. These two results have previously been used and derived by Cochrane and Orcutt (1949); Fröhlich (2017). The inverse matrix is given by

$$\boldsymbol{\Lambda}^{\mathrm{AR}} = (\boldsymbol{\Sigma}^{\mathrm{AR}})^{-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1} & & \\ & \ddots & \\ & & \boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{1} & -\phi\mathbf{1} & 0 & \cdots & 0 \\ -\phi\mathbf{1} & (1+\phi^2)\mathbf{1} & -\phi\mathbf{1} & \cdots & \vdots \\ 0 & -\phi\mathbf{1} & (1+\phi^2)\mathbf{1} & -\phi\mathbf{1} & \\ \vdots & & \ddots & \ddots & 0 \\ & & & (1+\phi^2)\mathbf{1} & -\phi\mathbf{1} \\ 0 & \cdots & 0 & -\phi\mathbf{1} & \mathbf{1} \end{bmatrix}. \tag{6.24}$$

This leads us to the normal equations of AR(1) NLLSQ estimation which are given by

$$\breve{\boldsymbol{J}}^{\top}\boldsymbol{\Lambda}^{\mathrm{AR}}\breve{\boldsymbol{J}}\Delta\boldsymbol{x}^{*} = -\breve{\boldsymbol{J}}^{\top}\boldsymbol{\Lambda}^{\mathrm{AR}}\breve{\boldsymbol{e}} \tag{6.25}$$

$$\Leftrightarrow \boldsymbol{H}^{\mathrm{AR}}\Delta\boldsymbol{x}^{*} = -\boldsymbol{b}^{\mathrm{AR}}. \tag{6.26}$$

This general description is valid for any kind of measurements with autocorrelated noise. In the following we analyze how constraints in the form of observed nodes and odometry edges influence the normal equations if they stem from those kind of measurements.

Observed nodes Let us examine the influence of observed nodes with autocorrelated noise on the estimation problem. Given the excerpt from a pose graph in Figure 6.8, we analyze the influence of two observed nodes. Let us assume that they both stem from the same pose source, that the underlying measurements share autocorrelated noise, and that they stem from two successive measurements. What kind of error do we make if we instead falsely assume that the constraints stem from a pose source with AWGN? To this end, we compute the changes in $\boldsymbol{H}$ and $\boldsymbol{b}$ when assuming either AWGN or AR(1) noise.

Assuming that the two observed nodes share autocorrelated noise, we find with (6.24)

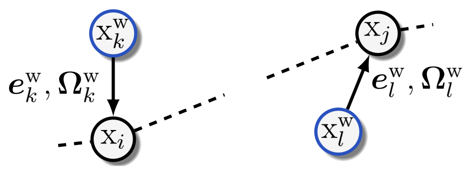


Figure 6.8.: Excerpt of a pose graph with two observed nodes. The dashed lines with the gap in the middle indicate that the hidden nodes x_i and x_j are part of the same graph but not necessarily directly connected.

that the information matrix for the two constraints is

$$\boldsymbol{\Lambda}^{w,AR} = \begin{bmatrix} \ddots & & & & \\ & \boldsymbol{\Lambda}_k^{w,AR} & \cdots & \boldsymbol{\Lambda}_{kl}^{w,AR} & \\ & \vdots & \ddots & \vdots & \\ & \boldsymbol{\Lambda}_{lk}^{w,AR} & \cdots & \boldsymbol{\Lambda}_l^{w,AR} & \\ & & & & \ddots \end{bmatrix}, \tag{6.27}$$

with

$$\boldsymbol{\Lambda}_k^{w,AR} = (1+\phi^2)\boldsymbol{\Lambda}_k^w, \tag{6.28}$$
$$\boldsymbol{\Lambda}_{kl}^{w,AR} = -\phi(\boldsymbol{\Lambda}_k^w)^{\frac{1}{2}}((\boldsymbol{\Lambda}_l^w)^{\frac{1}{2}})^\top, \tag{6.29}$$
$$\boldsymbol{\Lambda}_{lk}^{w,AR} = (\boldsymbol{\Lambda}_{kl}^{w,AR})^\top = \boldsymbol{\Lambda}_{kl}^{w,AR}, \tag{6.30}$$
$$\boldsymbol{\Lambda}_l^{w,AR} = (1+\phi^2)\boldsymbol{\Lambda}_l^w, \tag{6.31}$$

where $(\boldsymbol{\Lambda}_k^w)^{\frac{1}{2}}$ is the Cholesky decomposition. The structure of $\boldsymbol{\Lambda}^{w,AR}$ reflects that both constraints are somehow connected.

Let us denote $\breve{\boldsymbol{J}}_{k,i}^w = \left(\frac{\partial \boldsymbol{e}_k^w(\boldsymbol{x}_i, \boldsymbol{z}_k^w)}{\partial \boldsymbol{x}_i} \Big|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} \right)^\top$ and as usual stack all $\breve{\boldsymbol{J}}_{k,i}^w$ in $\breve{\boldsymbol{J}}^w$. With this notation we have

$$
\begin{aligned}
\boldsymbol{H}^{\mathrm{w,AR}} &= (\breve{\boldsymbol{J}}^{\mathrm{w}})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}} \breve{\boldsymbol{J}}^{\mathrm{w}} && (6.32)\\
&= \begin{bmatrix} \ddots & & & \\ & \boldsymbol{H}^{\mathrm{w,AR}}_{ii} & \cdots & \boldsymbol{H}^{\mathrm{w,AR}}_{ij} & \\ & \vdots & \ddots & \vdots & \\ & \boldsymbol{H}^{\mathrm{w,AR}}_{ji} & \cdots & \boldsymbol{H}^{\mathrm{w,AR}}_{jj} & \\ & & & & \ddots \end{bmatrix}, && (6.33)\\
&= \begin{bmatrix} \ddots & & & \\ & (\breve{\boldsymbol{J}}^{\mathrm{w}}_{k,i})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{k} \breve{\boldsymbol{J}}^{\mathrm{w}}_{k,i} & \cdots & \underline{(\breve{\boldsymbol{J}}^{\mathrm{w}}_{k,i})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{kl} \breve{\boldsymbol{J}}^{\mathrm{w}}_{l,j}} & \\ & \vdots & \ddots & \vdots & \\ & \underline{(\breve{\boldsymbol{J}}^{\mathrm{w}}_{l,j})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{kl} \breve{\boldsymbol{J}}^{\mathrm{w}}_{k,i}} & \cdots & (\breve{\boldsymbol{J}}^{\mathrm{w}}_{l,j})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{l} \breve{\boldsymbol{J}}^{\mathrm{w}}_{l,j} & \\ & & & & \ddots \end{bmatrix} && (6.34)
\end{aligned}
$$

and

$$
\boldsymbol{b}^{\mathrm{w,AR}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}^{\mathrm{w}}_{k,i})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{k} \breve{\boldsymbol{e}}^{\mathrm{w}}_{k} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{w}}_{k,i})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{kl} \breve{\boldsymbol{e}}^{\mathrm{w}}_{l}} \\ \vdots \\ (\breve{\boldsymbol{J}}^{\mathrm{w}}_{l,j})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{l} \breve{\boldsymbol{e}}^{\mathrm{w}}_{l} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{w}}_{l,j})^\top \boldsymbol{\Lambda}^{\mathrm{w,AR}}_{kl} \breve{\boldsymbol{e}}^{\mathrm{w}}_{k}} \\ \vdots \end{bmatrix}. \qquad (6.35)
$$

Here all terms underlined in blue are additional terms when comparing ordinary NLLSQ to AR(1) NLLSQ in the context of observed nodes. We see that these terms change the structure of $\boldsymbol{H}^{\mathrm{w,AR}}$ and decrease its sparsity. Additionally, we note that $\boldsymbol{H}^{\mathrm{w,AR}}_{ii}$ and $\boldsymbol{H}^{\mathrm{w,AR}}_{jj}$ differ in their values as we have to use a different information matrix as compared to when assuming AWGN. These differences are exactly the error that we make when we assume AWGN instead of autocorrelated noise.

In the next step, we are interested in a similar result for odometry edges.

Odometry edges Similarly to observed nodes, we are interested in knowing the error that we make when assuming AWGN instead of autocorrelated noise for odometry

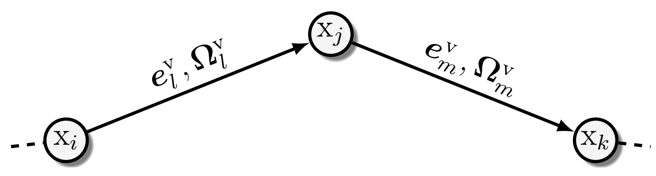

x_i
$\boldsymbol{e}_l^{\mathrm{v}}, \boldsymbol{\Omega}_l^{\mathrm{v}}$
x_j
$\boldsymbol{e}_m^{\mathrm{v}}, \boldsymbol{\Omega}_m^{\mathrm{v}}$
x_k

Figure 6.9.: Excerpt of a pose graph with two odometry edges.

constraints. In Figure 6.9 we depict a pose graph with two odometry constraints.

Again, we compute the resulting $\boldsymbol{H}$ and $\boldsymbol{b}$ by assuming that the two successive odometry constraints are autocorrelated. The stochastic model $\boldsymbol{\Lambda}^{\mathrm{v,AR}}$ is similar to the model for observed nodes:

$$\boldsymbol{\Lambda}^{\mathrm{v,AR}} = \begin{bmatrix} \ddots & & & & \\ & \boldsymbol{\Lambda}_l^{\mathrm{v,AR}} & \cdots & \boldsymbol{\Lambda}_{lm}^{\mathrm{v,AR}} & \\ & \vdots & \ddots & \vdots & \\ & \boldsymbol{\Lambda}_{ml}^{\mathrm{v,AR}} & \cdots & \boldsymbol{\Lambda}_m^{\mathrm{v,AR}} & \\ & & & & \ddots \end{bmatrix}, \tag{6.36}$$

with

$$\boldsymbol{\Lambda}_l^{\mathrm{v,AR}} = (1+\phi^2)\boldsymbol{\Lambda}_l^{\mathrm{v}}, \tag{6.37}$$

$$\boldsymbol{\Lambda}_{lm}^{\mathrm{v,AR}} = -\phi(\boldsymbol{\Lambda}_l^{\mathrm{v}})^{\frac{1}{2}}((\boldsymbol{\Lambda}_m^{\mathrm{v}})^{\frac{1}{2}})^{\top}, \tag{6.38}$$

$$\boldsymbol{\Lambda}_{ml}^{\mathrm{v,AR}} = (\boldsymbol{\Lambda}_{lm}^{\mathrm{v,AR}})^{\top} = \boldsymbol{\Lambda}_{lm}^{\mathrm{v,AR}}, \tag{6.39}$$

$$\boldsymbol{\Lambda}_m^{\mathrm{v,AR}} = (1+\phi^2)\boldsymbol{\Lambda}_m^{\mathrm{v}}. \tag{6.40}$$

The Jacobians, however, contain more terms. Therefore, we detail them to

$$\breve{\boldsymbol{J}}_l^{\mathrm{v}} = \begin{bmatrix} \cdots & \mathbf{0} & \overset{i}{\left.\frac{\partial \boldsymbol{e}_l^{\mathrm{v}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_l^{\mathrm{v}})}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}}} & \mathbf{0} & \cdots & \mathbf{0} & \overset{j}{\left.\frac{\partial \boldsymbol{e}_m^{\mathrm{v}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_l^{\mathrm{v}})}{\partial \boldsymbol{x}_j}\right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}}} & \mathbf{0} & \cdots \end{bmatrix} \tag{6.41}$$

$$= \begin{bmatrix} \cdots & \mathbf{0} & \breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}} & \mathbf{0} & \cdots & \mathbf{0} & \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}} & \mathbf{0} & \cdots \end{bmatrix}. \tag{6.42}$$

Similarly, we derive $\breve{\boldsymbol{J}}_m^{\mathrm{v}}$ by exchanging the subscript l to m. Stacking these two Jacobians leads to the matrix $\breve{\boldsymbol{J}}^{\mathrm{v}}$ with

$$\breve{\boldsymbol{J}}^{\mathrm{v}} = \begin{bmatrix} \vdots \\ \breve{\boldsymbol{J}}_l^{\mathrm{v}} \\ \vdots \\ \breve{\boldsymbol{J}}_m^{\mathrm{v}} \\ \vdots \end{bmatrix} = \begin{array}{c} \begin{array}{ccccccc} & i & & j & & k & \end{array} \\ \begin{bmatrix} & & & \vdots & & & \\ \cdots & \breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}} & \cdots & \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}} & \cdots & & \\ & & & \vdots & & & \\ & & \cdots & \breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}} & \cdots & \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}} & \cdots \\ & & & \vdots & & & \end{bmatrix} \end{array}. \tag{6.43}$$

Now we have the tools at hand to derive $\boldsymbol{H}^{\mathrm{v,AR}}$ to

$$\boldsymbol{H}^{\mathrm{v,AR}} = (\breve{\boldsymbol{J}}^{\mathrm{v}})^{\top} \boldsymbol{\Lambda}^{\mathrm{v,AR}} \breve{\boldsymbol{J}}^{\mathrm{v}} \tag{6.44}$$

$$= \begin{bmatrix} \ddots & & & & & & \\ & \boldsymbol{H}_{ii}^{\mathrm{v,AR}} & \cdots & \boldsymbol{H}_{ij}^{\mathrm{v,AR}} & \cdots & \boldsymbol{H}_{ik}^{\mathrm{v,AR}} & \\ & \vdots & & \vdots & & \vdots & \\ & \boldsymbol{H}_{ji}^{\mathrm{v,AR}} & \cdots & \boldsymbol{H}_{jj}^{\mathrm{v,AR}} & \cdots & \boldsymbol{H}_{jk}^{\mathrm{v,AR}} & \\ & \vdots & & \vdots & & \vdots & \\ & \boldsymbol{H}_{ki}^{\mathrm{v,AR}} & \cdots & \boldsymbol{H}_{kj}^{\mathrm{v,AR}} & \cdots & \boldsymbol{H}_{kk}^{\mathrm{v,AR}} & \\ & & & & & & \ddots \end{bmatrix}, \tag{6.45}$$

with

$$\boldsymbol{H}^{\mathrm{v,AR}}_{ii} = (\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,i})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{l} \breve{\boldsymbol{J}}^{\mathrm{v}}_{l,i}, \tag{6.46}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{ij} = (\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,i})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{l} \breve{\boldsymbol{J}}^{\mathrm{v}}_{l,j} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,i})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{J}}^{\mathrm{v}}_{m,j}}, \tag{6.47}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{ik} = \underline{(\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,i})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{J}}^{\mathrm{v}}_{m,k}}, \tag{6.48}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{ji} = (\boldsymbol{H}^{\mathrm{v,AR}}_{ij})^\top, \tag{6.49}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{jj} = (\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,j})^\top (\boldsymbol{\Lambda}^{\mathrm{v,AR}}_{l} \breve{\boldsymbol{J}}^{\mathrm{v}}_{l,j} + \underline{\boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{J}}^{\mathrm{v}}_{m,j}}) + (\breve{\boldsymbol{J}}^{\mathrm{v}}_{m,j})^\top (\boldsymbol{\Lambda}^{\mathrm{v,AR}}_{m} \breve{\boldsymbol{J}}^{\mathrm{v}}_{m,j} + \underline{\boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{J}}^{\mathrm{v}}_{l,j}}), \tag{6.50}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{jk} = (\breve{\boldsymbol{J}}^{\mathrm{v}}_{m,j})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{m} \breve{\boldsymbol{J}}^{\mathrm{v}}_{m,k} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,j})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{J}}^{\mathrm{v}}_{m,k}}, \tag{6.51}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{ki} = (\boldsymbol{H}^{\mathrm{v,AR}}_{ik})^\top, \tag{6.52}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{kj} = (\boldsymbol{H}^{\mathrm{v,AR}}_{jk})^\top, \tag{6.53}$$

$$\boldsymbol{H}^{\mathrm{v,AR}}_{kk} = (\breve{\boldsymbol{J}}^{\mathrm{v}}_{m,k})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{m} \breve{\boldsymbol{J}}^{\mathrm{v}}_{m,k}. \tag{6.54}$$

Similarly, we obtain $\boldsymbol{b}^{\mathrm{v,AR}}$ with

$$\boldsymbol{b}^{\mathrm{v,AR}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,i})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{l} \breve{\boldsymbol{e}}^{\mathrm{v}}_{l} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,i})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{e}}^{\mathrm{v}}_{m}} \\ \vdots \\ (\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,j})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{l} \breve{\boldsymbol{e}}^{\mathrm{v}}_{l} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{v}}_{l,j})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{e}}^{\mathrm{v}}_{m}} + (\breve{\boldsymbol{J}}^{\mathrm{v}}_{m,j})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{m} \breve{\boldsymbol{e}}^{\mathrm{v}}_{m} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{v}}_{m,j})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{e}}^{\mathrm{v}}_{l}} \\ \vdots \\ (\breve{\boldsymbol{J}}^{\mathrm{v}}_{m,k})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{m} \breve{\boldsymbol{e}}^{\mathrm{v}}_{m} + \underline{(\breve{\boldsymbol{J}}^{\mathrm{v}}_{m,k})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}}_{lm} \breve{\boldsymbol{e}}^{\mathrm{v}}_{l}} \\ \vdots \end{bmatrix}. \tag{6.55}$$

Again, all terms underlined in blue are additional terms compared to ordinary NLLSQ, this time in the context of odometry edges. Also, all information matrices are different. Therefore, we can specify the error that we make when assuming AWGN for odometry constraints that in fact have autocorrelated noise.

In conclusion, these derivations for observed nodes and odometry edges provide us with a better stochastic model for constraints with autocorrelated noise. Therefore, the resulting uncertainty estimation of the result is of higher quality. However, we have also seen that the augmented optimization problem is harder to solve as it becomes less sparse and more memory consuming. As a consequence, if we would implement these models within the core estimator, we would lose its block-tridiagonal structure and the associated advantages. Moreover, we have derived the AR(1) NLLSQ estimation procedure, but we do not have a graph-based representation for it. We are generally interested in such a representation to help us reason more intuitively about the underlying problem structure. For this reason we have already derived in Section 5.3 a prior node to represent the effect of marginalization on our sliding window pose graph. In the same spirit we derive graph elements to represent constraints with autocorrelated noise in the following.

6.4.2. Constraints with autocorrelated noise can be understood with graph elements

In Section 6.4.1 we show how the AR(1) NLLSQ estimation looks like. Now we derive an understanding of this in terms of the graph-based representation. For this our current tool set of error functions for a pose graph, $\boldsymbol{e}^{\mathrm{v}}$ and $\boldsymbol{e}^{\mathrm{w}}$, is not sufficient anymore. We will see that we need to add edges with additional error functions. As a first step, we derive edges that model observed nodes with autocorrelated noise. As a second step, we derive similar edges that model odometry edges with autocorrelated noise. Together, they allow us to create a graph-based representation of a AR(1) NLLSQ problem.

Observed nodes Our goal is to derive edges that model the influence of observed nodes with autocorrelated noise. This influence should be the same as derived in Section 6.4.1. We approach this by analyzing two observed nodes with autocorrelated noise, detailing the corresponding edges, and proving that they lead to the desired result.

At first, we analyze two observed nodes $\mathrm{x}_k^{\mathrm{w}}$ and $\mathrm{x}_l^{\mathrm{w}}$ with the pose estimates $\boldsymbol{z}_k^{\mathrm{w}}$ and $\boldsymbol{z}_l^{\mathrm{w}}$. Let them be connected to two hidden nodes x_i and x_j, as depicted in Figure 6.8. Under the AWGN assumption they lead to addends in $\boldsymbol{H}_{ii}, \boldsymbol{H}_{jj}, \boldsymbol{b}_i$, and $\boldsymbol{b}_j$ (cf. Section 3.4.6). However, if we assume their noise to be modeled as AR(1), then they should lead to different addends in these places and additionally to addends in $\boldsymbol{H}_{ij}$ and $\boldsymbol{H}_{ji}$. What graph elements do we need to achieve this?

We combine three edges $\mathrm{z}_k^{\mathrm{w}}, \mathrm{z}_l^{\mathrm{w}}$, and $\mathrm{z}_m^{\mathrm{w,AR}}$ to achieve this result. The Figure 6.10

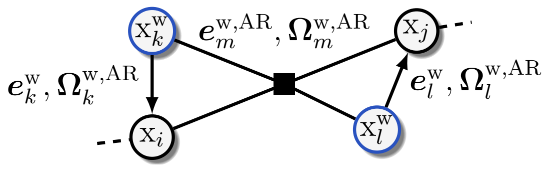

$\mathrm{x}_k^{\mathrm{w}}$
$\boldsymbol{e}_m^{\mathrm{w,AR}}, \boldsymbol{\Omega}_m^{\mathrm{w,AR}}$
x_j
$\boldsymbol{e}_k^{\mathrm{w}}, \boldsymbol{\Omega}_k^{\mathrm{w,AR}}$
$\boldsymbol{e}_l^{\mathrm{w}}, \boldsymbol{\Omega}_l^{\mathrm{w,AR}}$
x_i
$\mathrm{x}_l^{\mathrm{w}}$

Figure 6.10.: The same part of a pose graph as in Figure 6.8, but this time we assume the noise of the observed nodes to be modeled as AR(1). We need an additional constraint to model the influence of the autocorrelated noise.

shows the same part of the graph as Figure 6.8, but this time we assume the noise of the observed nodes to be described as AR(1). The most striking difference when comparing this to the graph in Figure 6.8 is the introduction of the hyperedge $\mathrm{z}_m^{\mathrm{w,AR}}$ that connects both observed nodes and both hidden nodes simultaneously. Looking more closely, we also see that the edges from the observed to the hidden nodes have undergone a correction of their associated information matrices. We derive in the following that these three edges have the same influence on the underlying NLLSQ formulation as directly augmenting the stochastic model in the optimization problem as we have done in Section 6.4.1. We conclude that these edges are therefore the graph-based representation of constraints from measurements with autocorrelated noise.

The new hyperedge $\mathrm{z}_m^{\mathrm{w,AR}}$ connects $\mathrm{x}_k^{\mathrm{w}}, \mathrm{x}_l^{\mathrm{w}}, \mathrm{x}_i$, and x_j. It depends on a new error function $\boldsymbol{e}_m^{\mathrm{w,AR}}$ with

$$\boldsymbol{e}_m^{\mathrm{w,AR}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{w}}, \boldsymbol{z}_l^{\mathrm{w}}) = \boldsymbol{e}_k^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}}) + \boldsymbol{e}_l^{\mathrm{w}}(\boldsymbol{x}_j, \boldsymbol{z}_l^{\mathrm{w}}). \tag{6.56}$$

Its partial derivatives are

$$\frac{\partial \boldsymbol{e}_m^{\mathrm{w,AR}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{w}}, \boldsymbol{z}_l^{\mathrm{w}})}{\partial \boldsymbol{x}_i} = \frac{\partial \boldsymbol{e}_k^{\mathrm{w}}(\boldsymbol{x}_i, \boldsymbol{z}_k^{\mathrm{w}})}{\partial \boldsymbol{x}_i} = \breve{\boldsymbol{J}}_{m,i}^{\mathrm{w,AR}} = \breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}}, \tag{6.57}$$

$$\frac{\partial \boldsymbol{e}_m^{\mathrm{w,AR}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_k^{\mathrm{w}}, \boldsymbol{z}_l^{\mathrm{w}})}{\partial \boldsymbol{x}_j} = \frac{\partial \boldsymbol{e}_l^{\mathrm{w}}(\boldsymbol{x}_j, \boldsymbol{z}_l^{\mathrm{w}})}{\partial \boldsymbol{x}_j} = \breve{\boldsymbol{J}}_{m,j}^{\mathrm{w,AR}} = \breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}}. \tag{6.58}$$

Its Jacobian is therefore

$$\breve{\boldsymbol{J}}_m^{\mathrm{w,AR}} = \begin{array}{c} \begin{array}{ccccccccc} & & i & & & & j & & \\ \end{array} \\ \left[\begin{array}{ccccccccc} \cdots & \mathbf{0} & \left.\frac{\partial \boldsymbol{e}_m^{\mathrm{w,AR}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_k^{\mathrm{w}},\boldsymbol{z}_l^{\mathrm{w}})}{\partial \boldsymbol{x}_i}\right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} & \mathbf{0} & \cdots & \mathbf{0} & \left.\frac{\partial \boldsymbol{e}_m^{\mathrm{w,AR}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_k^{\mathrm{w}},\boldsymbol{z}_l^{\mathrm{w}})}{\partial \boldsymbol{x}_j}\right|_{\boldsymbol{x}=\breve{\boldsymbol{x}}} & \mathbf{0} & \cdots \end{array}\right] \end{array} \tag{6.59}$$

$$= \begin{array}{c} \begin{array}{ccccccccc} & & i & & & & j & & \end{array} \\ \left[\begin{array}{ccccccccc} \cdots & \mathbf{0} & \breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}} & \mathbf{0} & \cdots & \mathbf{0} & \breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}} & \mathbf{0} & \cdots \end{array}\right] \end{array}. \tag{6.60}$$

The corresponding constraint $\mathrm{z}_m^{\mathrm{w,AR}}$ takes into account this error function and the information matrix $\boldsymbol{\Lambda}_m^{\mathrm{w,AR}} = \boldsymbol{\Lambda}_{kl}^{\mathrm{w,AR}}$ such that the squared error becomes

$$e_m^{\mathrm{w,AR}} = \boldsymbol{e}_m^{\mathrm{w,AR}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_k^{\mathrm{w}},\boldsymbol{z}_l^{\mathrm{w}})^\top \boldsymbol{\Lambda}_m^{\mathrm{w,AR}} \boldsymbol{e}_m^{\mathrm{w,AR}}(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{z}_k^{\mathrm{w}},\boldsymbol{z}_l^{\mathrm{w}}). \tag{6.61}$$

This constraint leads to $\boldsymbol{H}_m^{\mathrm{w,AR}}$ with

$$\boldsymbol{H}_m^{\mathrm{w,AR}} = \begin{bmatrix} \ddots & & & & \\ & (\breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_m^{\mathrm{w,AR}} \breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}} & \cdots & (\breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_m^{\mathrm{w,AR}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}} & \\ & \vdots & \ddots & \vdots & \\ & (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_m^{\mathrm{w,AR}} \breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}} & \cdots & (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_m^{\mathrm{w,AR}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}} & \\ & & & & \ddots \end{bmatrix}, \tag{6.62}$$

and $\boldsymbol{b}_m^{\mathrm{w,AR}}$ with

$$\boldsymbol{b}_m^{\mathrm{w,AR}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_m^{\mathrm{w,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{w,AR}} \\ \vdots \\ (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_m^{\mathrm{w,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{w,AR}} \\ \vdots \end{bmatrix}. \tag{6.63}$$

This is a nice intermediate result as it leads to the correct addends in $\boldsymbol{H}_{ij}^{\mathrm{w,AR}}$ and $\boldsymbol{H}_{ji}^{\mathrm{w,AR}}$, but because of the information matrix $\boldsymbol{\Lambda}_m^{\mathrm{w,AR}}$ it leads to slightly wrong addends in $\boldsymbol{H}_{ii}^{\mathrm{w,AR}}$, $\boldsymbol{H}_{jj}^{\mathrm{w,AR}}$, $\boldsymbol{b}_i^{\mathrm{w,AR}}$, and $\boldsymbol{b}_j^{\mathrm{w,AR}}$. We correct for this by considering the other two mentioned edges $\mathrm{z}_k^{\mathrm{w}}$ and $\mathrm{z}_l^{\mathrm{w}}$. In terms of the error functions, we use them as ordinary edges between observed and hidden nodes, just as we would model them when assuming AWGN noise. However, we adapt the associated information matrices to be

$$\boldsymbol{\Lambda}_k^{\mathrm{w}} = \boldsymbol{\Lambda}_k^{\mathrm{w,AR}} - \boldsymbol{\Lambda}_m^{\mathrm{w,AR}}, \tag{6.64}$$

$$\boldsymbol{\Lambda}_l^{\mathrm{w}} = \boldsymbol{\Lambda}_l^{\mathrm{w,AR}} - \boldsymbol{\Lambda}_m^{\mathrm{w,AR}}. \tag{6.65}$$

The two constraints $\mathrm{z}_k^{\mathrm{w}}$ and $\mathrm{z}_l^{\mathrm{w}}$ lead with these corrected information matrices to $\boldsymbol{H}_k^{\mathrm{w}}$ and $\boldsymbol{H}_l^{\mathrm{w}}$ with

$$\boldsymbol{H}_k^{\mathrm{w}} = \begin{bmatrix} \ddots & & \\ & (\breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_k^{\mathrm{w}} \breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}} & \\ & & \ddots \end{bmatrix}, \tag{6.66}$$

$$\boldsymbol{H}_l^{\mathrm{w}} = \begin{bmatrix} \ddots & & \\ & (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_l^{\mathrm{w}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{w}} & \\ & & \ddots \end{bmatrix}. \tag{6.67}$$

Equivalently we find that

$$\boldsymbol{b}_k^{\mathrm{w}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_k^{\mathrm{w}} \breve{\boldsymbol{e}}_k^{\mathrm{w}} \\ \vdots \end{bmatrix}, \tag{6.68}$$

$$\boldsymbol{b}_l^{\mathrm{w}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}_{k,i}^{\mathrm{w}})^\top \boldsymbol{\Lambda}_l^{\mathrm{w}} \breve{\boldsymbol{e}}_l^{\mathrm{w}} \\ \vdots \end{bmatrix}. \tag{6.69}$$

These three constraints together allow us to compute $\boldsymbol{H}^{\mathrm{w,AR}}$ and $\boldsymbol{b}^{\mathrm{w,AR}}$ for the graph depicted in Figure 6.10:

$$\boldsymbol{H}^{\mathrm{w,AR}} = \boldsymbol{H}_k^{\mathrm{w}} + \boldsymbol{H}_l^{\mathrm{w}} + \boldsymbol{H}_m^{\mathrm{w,AR}}, \tag{6.70}$$

$$\boldsymbol{b}^{\mathrm{w,AR}} = \boldsymbol{b}_k^{\mathrm{w}} + \boldsymbol{b}_l^{\mathrm{w}} + \boldsymbol{b}_m^{\mathrm{w,AR}}. \tag{6.71}$$

$$\tag{6.72}$$

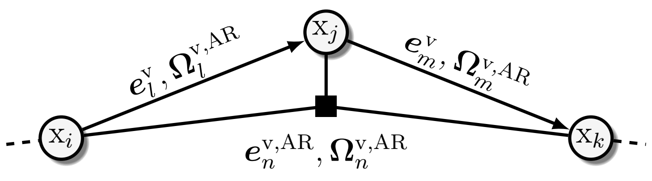

$\boldsymbol{e}_l^{\mathrm{v}}, \boldsymbol{\Omega}_l^{\mathrm{v,AR}}$
x_j
$\boldsymbol{e}_m^{\mathrm{v}}, \boldsymbol{\Omega}_m^{\mathrm{v,AR}}$
x_i
$\boldsymbol{e}_n^{\mathrm{v,AR}}, \boldsymbol{\Omega}_n^{\mathrm{v,AR}}$
x_k

Figure 6.11.: The same part of a pose graph as in Figure 6.9, but this time we assume the noise of the odometry edges to be modeled as AR(1). We need an additional constraint to model the influence of the noise of the AR(1) process.

It is straightforward to verify that these results are equal to (6.34) and (6.35). Therefore, we have defined a graph-based representation for modeling observed nodes with autocorrelated noise that is equal to the direct modeling as a AR(1) NLLSQ problem. A key piece that we are still missing is, how a graph-based representation of odometry edges with autocorrelated noise looks like. We approach this question in the following.

Odometry edges Within a similar train of thought as the derivation of a graph-based representation for observed nodes with autocorrelated noise, it is now our goal to derive the same for odometry edges. To this end, we begin by stating that the graph-based representation in Figure 6.11 solves this problem. With a similar approach as for observed nodes, we first define the new hyperedge $\mathrm{z}_n^{\mathrm{v,AR}}$ and then show that we need to adapt the information matrices of the usual constraints $\mathrm{z}_l^{\mathrm{v}}$ and $\mathrm{z}_m^{\mathrm{v}}$ to achieve the desired result.

The constraint $\mathrm{z}_n^{\mathrm{v,AR}}$ of the corresponding hyperedge is associated to the error function

$$\boldsymbol{e}_n^{\mathrm{v,AR}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{z}_l^{\mathrm{v}}, \boldsymbol{z}_m^{\mathrm{v}}) = \boldsymbol{e}_l^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_l^{\mathrm{v}}) + \boldsymbol{e}_m^{\mathrm{v}}(\boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{z}_m^{\mathrm{v}}). \tag{6.73}$$

Its partial derivatives with respect to the state variables are

$$\frac{\partial \boldsymbol{e}_n^{\mathrm{v,AR}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{z}_l^{\mathrm{v}}, \boldsymbol{z}_m^{\mathrm{v}})}{\partial \boldsymbol{x}_i} = \frac{\partial \boldsymbol{e}_l^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_l^{\mathrm{w}})}{\partial \boldsymbol{x}_i}, \tag{6.74}$$

$$\frac{\partial \boldsymbol{e}_n^{\mathrm{v,AR}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{z}_l^{\mathrm{v}}, \boldsymbol{z}_m^{\mathrm{v}})}{\partial \boldsymbol{x}_j} = \frac{\partial \boldsymbol{e}_l^{\mathrm{v}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_l^{\mathrm{w}})}{\partial \boldsymbol{x}_j} + \frac{\partial \boldsymbol{e}_m^{\mathrm{v}}(\boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{z}_m^{\mathrm{w}})}{\partial \boldsymbol{x}_j}, \tag{6.75}$$

$$\frac{\partial \boldsymbol{e}_n^{\mathrm{v,AR}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{z}_l^{\mathrm{v}}, \boldsymbol{z}_m^{\mathrm{v}})}{\partial \boldsymbol{x}_k} = \frac{\partial \boldsymbol{e}_m^{\mathrm{v}}(\boldsymbol{x}_j, \boldsymbol{x}_k, \boldsymbol{z}_m^{\mathrm{w}})}{\partial \boldsymbol{x}_k}. \tag{6.76}$$

With the usual abbreviations the partial derivatives evaluated at the linearization point are

$$\breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}} = \breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}}, \tag{6.77}$$
$$\breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}} = \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}} + \breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}}, \tag{6.78}$$
$$\breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}} = \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}}. \tag{6.79}$$

The Jacobian of the constraint $\mathrm{z}_n^{\mathrm{v,AR}}$ is therefore

$$\breve{\boldsymbol{J}}_n^{\mathrm{v,AR}} = \begin{array}{c} \begin{array}{ccccccccccccc} & & i & & & & j & & & & k & & \\ \cdots & \mathbf{0} & \breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}} & \mathbf{0} & \cdots & \mathbf{0} & \breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}} & \mathbf{0} & \cdots & \mathbf{0} & \breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}} & \mathbf{0} & \cdots \end{array} \end{array}. \tag{6.80}$$

The constraint $\mathrm{z}_n^{\mathrm{v,AR}}$ leads to $\boldsymbol{H}_n^{\mathrm{v,AR}}$ with

$$\boldsymbol{H}_n^{\mathrm{v,AR}} = (\breve{\boldsymbol{J}}_n^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_n^{\mathrm{v,AR}} = \tag{6.81}$$

$$\begin{bmatrix} \ddots & & & & & & \\ & (\breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}} & \cdots & (\breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}} & \cdots & (\breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}} & \\ & \vdots & & \vdots & & \vdots & \\ & (\breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}} & \cdots & (\breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}} & \cdots & (\breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}} & \\ & \vdots & & \vdots & & \vdots & \\ & (\breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}} & \cdots & (\breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}} & \cdots & (\breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}} & \\ & & & & & & \ddots \end{bmatrix} \tag{6.82}$$

and $\boldsymbol{b}_n^{\mathrm{v,AR}}$ with

$$\boldsymbol{b}_n^{\mathrm{v,AR}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}_{n,i}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_n^{\mathrm{v,AR}} \\ \vdots \\ (\breve{\boldsymbol{J}}_{n,j}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_n^{\mathrm{v,AR}} \\ \vdots \\ (\breve{\boldsymbol{J}}_{n,k}^{\mathrm{v,AR}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_n^{\mathrm{v,AR}} \\ \vdots \end{bmatrix}. \tag{6.83}$$

Using the definition of the Jacobians we develop (6.82) to $\boldsymbol{H}_n^{\mathrm{v,AR}}$ with the entries

$$\boldsymbol{H}_{n,ii}^{\mathrm{v,AR}} = \underline{(\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}}}, \tag{6.84}$$

$$\boldsymbol{H}_{n,ij}^{\mathrm{v,AR}} = \underline{(\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}}} + (\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}}, \tag{6.85}$$

$$\boldsymbol{H}_{n,ik}^{\mathrm{v,AR}} = (\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}}, \tag{6.86}$$

$$\boldsymbol{H}_{n,ji}^{\mathrm{v,AR}} = (\boldsymbol{H}_{n,ij}^{\mathrm{v,AR}})^\top, \tag{6.87}$$

$$\boldsymbol{H}_{n,jj}^{\mathrm{v,AR}} = (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}})^\top (\underline{\boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}}} + \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}}) + (\breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}})^\top (\underline{\boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}}} + \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}}), \tag{6.88}$$

$$\boldsymbol{H}_{n,jk}^{\mathrm{v,AR}} = \underline{(\breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}}} + (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}}, \tag{6.89}$$

$$\boldsymbol{H}_{n,ki}^{\mathrm{v,AR}} = (\boldsymbol{H}_{n,ik}^{\mathrm{v,AR}})^\top, \tag{6.90}$$

$$\boldsymbol{H}_{n,kj}^{\mathrm{v,AR}} = (\boldsymbol{H}_{n,jk}^{\mathrm{v,AR}})^\top, \tag{6.91}$$

$$\boldsymbol{H}_{n,kk}^{\mathrm{v,AR}} = \underline{(\breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}}} \tag{6.92}$$

and similarly, (6.83) to

$$\boldsymbol{b}_n^{\mathrm{v,AR}} = \begin{bmatrix} \vdots \\ \underline{(\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_l^{\mathrm{v}}} + (\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{v}} \\ \vdots \\ \underline{(\breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_l^{\mathrm{v}}} + (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{v}} + \underline{(\breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{v}}} + (\breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_l^{\mathrm{v}} \\ \vdots \\ \underline{(\breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{v}}} + (\breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_n^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_l^{\mathrm{v}} \\ \vdots \end{bmatrix}. \tag{6.93}$$

As in the derivation for the graph-based representation of observed nodes with auto-correlated noise, we notice that some of these matrix entries are already correct, while others need some modification. To be more precise, the entries underlined in blue are not yet identical when comparing them to (6.45) and (6.55), which is our goal. This is caused by the deviation of the information matrices. We correct for this by adapting the information matrices of the constraints $\mathrm{z}_l^{\mathrm{v}}$ and $\mathrm{z}_m^{\mathrm{v}}$ to

$$\boldsymbol{\Lambda}_l^{\mathrm{v}} = \boldsymbol{\Lambda}_l^{\mathrm{v,AR}} - \boldsymbol{\Lambda}_n^{\mathrm{v,AR}}, \tag{6.94}$$

$$\boldsymbol{\Lambda}_m^{\mathrm{v}} = \boldsymbol{\Lambda}_m^{\mathrm{v,AR}} - \boldsymbol{\Lambda}_n^{\mathrm{v,AR}}. \tag{6.95}$$

These two constraints then lead to $\boldsymbol{H}_l^{\mathrm{v}}$ and $\boldsymbol{H}_m^{\mathrm{v}}$ with

$$\boldsymbol{H}_l^{\mathrm{v}} = \begin{bmatrix} \ddots & & & & \\ & (\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_l^{\mathrm{v}} \breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}} & \cdots & (\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_l^{\mathrm{v}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}} & \\ & \vdots & & \vdots & \\ & (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_l^{\mathrm{v}} \breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}} & \cdots & (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_l^{\mathrm{v}} \breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}} & \\ & & & & \ddots \end{bmatrix} \tag{6.96}$$

and

$$\boldsymbol{H}_m^{\mathrm{v}} = \begin{bmatrix} \ddots & & & & \\ & (\breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_m^{\mathrm{v}} \breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}} & \cdots & (\breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_m^{\mathrm{v}} \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}} & \\ & \vdots & & \vdots & \\ & (\breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_m^{\mathrm{v}} \breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}} & \cdots & (\breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_m^{\mathrm{v}} \breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}} & \\ & & & & \ddots \end{bmatrix}. \tag{6.97}$$

The corresponding right-hand side vectors are $\boldsymbol{b}_l^{\mathrm{v}}$ and $\boldsymbol{b}_m^{\mathrm{v}}$ with

$$\boldsymbol{b}_l^{\mathrm{v}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}_{l,i}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_l^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_l^{\mathrm{v}} \\ \vdots \\ (\breve{\boldsymbol{J}}_{l,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_l^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_l^{\mathrm{v}} \\ \vdots \end{bmatrix}, \tag{6.98}$$

$$\boldsymbol{b}_m^{\mathrm{v}} = \begin{bmatrix} \vdots \\ (\breve{\boldsymbol{J}}_{m,j}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_m^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{v}} \\ \vdots \\ (\breve{\boldsymbol{J}}_{m,k}^{\mathrm{v}})^\top \boldsymbol{\Lambda}_m^{\mathrm{v,AR}} \breve{\boldsymbol{e}}_m^{\mathrm{v}} \\ \vdots \end{bmatrix}. \tag{6.99}$$

In total, the three constraints $\mathrm{z}_l^{\mathrm{v}}$, $\mathrm{z}_m^{\mathrm{v}}$, and $\mathrm{z}_n^{\mathrm{v,AR}}$ allow us to compute $\boldsymbol{H}^{\mathrm{v,AR}}$ and $\boldsymbol{b}^{\mathrm{v,AR}}$ for the graph depicted in Figure 6.11 to

$$\boldsymbol{H}^{\mathrm{v,AR}} = \boldsymbol{H}_l^{\mathrm{v}} + \boldsymbol{H}_m^{\mathrm{v}} + \boldsymbol{H}_n^{\mathrm{v,AR}}, \tag{6.100}$$

$$\boldsymbol{b}^{\mathrm{v,AR}} = \boldsymbol{b}_l^{\mathrm{v}} + \boldsymbol{b}_m^{\mathrm{v}} + \boldsymbol{b}_n^{\mathrm{v,AR}}. \tag{6.101}$$

We compare the result with (6.45) and (6.55) and verify that they are indeed identical. We conclude that the given three constraints are a graph-based representation for odometry edges with autocorrelated noise. Together with the graph-based representation for observed nodes with autocorrelated noise, we are now able to understand the influence of this kind of noise on the graph-based representation. This by itself is a big step forward. However, as these new constraints lead to the exact same entries as AR(1) NLLSQ, we still suffer from the increased computational demand. In the next section we therefore tackle this challenge.

6.4.3. AR(1) scaling: efficient implementation by scaling information matrices

In the previous section we described how graph elements can represent constraints with autocorrelated errors. The upside of this is that we gain understanding in terms of how the graphical structure is influenced by these constraints. However, the AR(1) NLLSQ scheme results in a decreased sparsity of the system matrix compared to ordinary NLLSQ. The graph elements for constraints with autocorrelated errors share this property of course, as they provide an exact model of the underlying equations. A reduced sparsity generally means that solving the linear system takes more time. In this section we present a remedy for this which allows us to retain the same sparsity pattern as in ordinary NLLSQ. In fact, we will see that we only need to inflate the covariances matrices of the measurements in a specific way to obtain a covariance consistent estimate. We refer to this as *AR(1) scaling*.

As discussed earlier, performing ordinary NLLSQ estimation on constraints with autocorrelated errors leads to unbiased mean estimates but to overconfident uncertainty estimates. Therefore, we explicitly focus on how to obtain better uncertainty estimates. To this end, we consider the information-theoretic optimal fusion of measurements with autocorrelated errors by computing the FI matrix of AR(1) NLLSQ. Similar to CI, we postulate that the information content of our scaled covariance matrices has to be equal to that of the FI matrix of the measurements.

Let Z_i be the random variable that describes the measurements $\boldsymbol{z}_i^{\mathrm{AR}}$. For the computation of the FI matrix $\boldsymbol{I}(Z|\boldsymbol{\mu})$, we need the joint probability distribution of the n random variables $Z = (Z_1, \dots, Z_n)$ with

$$p(Z) \propto \exp\left(-\frac{1}{2}[Z_1, \dots, Z_n](\boldsymbol{\Sigma}^{\mathrm{AR}})^{-1}[Z_1, \dots, Z_n]^\top\right). \tag{6.102}$$

At this point we make use of our derivation in Section 3.7.1 where we derived the CRLB applied to measurements from independent Gaussians. The key difference is that we do not assume independence between the measurements anymore. This changes how the matrix $\boldsymbol{B}$ looks like. For measurements with autocorrelated errors, we have $\boldsymbol{B} = \boldsymbol{\Sigma}^{\mathrm{AR}}$. Keeping this difference in mind, we follow the derivation from (3.99) to

(3.104) and conclude that

$$\boldsymbol{I}(Z|\boldsymbol{\mu}) = \boldsymbol{A}^\top(\boldsymbol{\Sigma}^{\mathrm{AR}})^{-1}\boldsymbol{A}, \qquad \boldsymbol{A} = \underbrace{[\mathbf{1},\dots,\mathbf{1}]^\top}_{n \text{ blocks}}. \tag{6.103}$$

The FI matrix is therefore the sum of all entries of $(\boldsymbol{\Sigma}^{\mathrm{AR}})^{-1}$ such that

$$\boldsymbol{I}(Z|\boldsymbol{\mu}) = \sum_{i,j}(\boldsymbol{\Sigma}^{\mathrm{AR}}_{ij})^{-1} = \sum_{i,j}\boldsymbol{\Lambda}^{\mathrm{AR}}_{ij}. \tag{6.104}$$

Luckily, we already know $\boldsymbol{\Sigma}^{\mathrm{AR}}$ and its inverse, $\boldsymbol{\Lambda}^{\mathrm{AR}}$, from (6.23) and (6.24). We can now plug in these terms and obtain

$$\begin{aligned}\boldsymbol{I}(Z|\boldsymbol{\mu}) &= \sum_{i,j}\boldsymbol{\Lambda}^{\mathrm{AR}}_{ij} & (6.105)\\ &= \boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1}(\underbrace{(n-2)(1+\phi^2)}_{\text{diagonal entries}} - \underbrace{2(n-1)\phi}_{\text{off-diagonal entries}} + \underbrace{2}_{\text{first and last entry}}) & (6.106)\\ &= \boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1}((n-2)\phi^2 - 2(n-1)\phi + n). & (6.107)\end{aligned}$$

We employ a similar weighting technique as CI for fusing measurements with autocorrelated noise. To this end, we weight the covariance matrix $\boldsymbol{\Sigma}^{\mathrm{AR}}_i$ of each measurement with a weight ω_i such that we obtain $\omega_i\boldsymbol{\Sigma}^{\mathrm{AR}}_i$ for each measurement. The key question is then how to choose the weights ω_i. CI proposes to chose the weights in a very conservative way such that $\sum_{i=1}^n \omega_i = 1$. As we have shown in Section 3.7.2, this means that we consider the information content of a *single* measurement because we do not want to make any assumption about the nature of the correlation. This is different for measurements with autocorrelated noise, where we have more knowledge about the correlation. We postulate that the information content of all measurements should be equal to the FI matrix of the measurements. This guarantees that we make optimal use of the information at hand while at the same time not being overly confident.

To formalize this, we postulate that

$$\sum_{i=1}^n \omega_i(\boldsymbol{\Sigma}^{\mathrm{AR}}_i)^{-1} = \boldsymbol{I}(Z|\boldsymbol{\mu}). \tag{6.108}$$

With (6.107), we extend this to

$$\sum_{i=1}^{n} \omega_i (\boldsymbol{\Sigma}_i^{\mathrm{AR}})^{-1} = \boldsymbol{I}(Z|\boldsymbol{\mu}) \tag{6.109}$$

$$\Leftrightarrow \quad \sum_{i=1}^{n} \omega_i \boldsymbol{\Lambda}_i^{\mathrm{AR}} = \boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1}((n-2)\phi^2 - 2(n-1)\phi + n) \tag{6.110}$$

$$\Leftrightarrow \quad \boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1}(1-\phi^2)\sum_{i=1}^{n} \omega_i = \boldsymbol{\Sigma}_{\boldsymbol{v}}^{-1}((n-2)\phi^2 - 2(n-1)\phi + n) \tag{6.111}$$

$$\Leftrightarrow \quad \sum_{i=1}^{n} \omega_i = \frac{(n-2)\phi^2 - 2(n-1)\phi + n}{1-\phi^2} \tag{6.112}$$

$$\Leftrightarrow \quad \sum_{i=1}^{n} \omega_i = \frac{n-(n-2)\phi}{1+\phi}. \tag{6.113}$$

If we assume that all weights are equal, then they can be computed by

$$\omega_i = \frac{n-(n-2)\phi}{n(1+\phi)}, \qquad i = 1, \ldots, n. \tag{6.114}$$

This closed-form solution is easily computable. This final result gives us a nice way of representing the information content of constraints with autocorrelated noise. It is also easily applicable and does not influence the sparsity of the optimization problem.

Lastly, we show that fusion without correlated constraints and CI are a special case of this scaling. We have the ordinary NLLSQ problem with $\phi = 0$ for which do not expect any scaled information matrices. Indeed, we find that

$$\omega_i = \frac{n}{n} = 1. \tag{6.115}$$

Applying CI is, as previously shown, equivalent to assuming the information content of a single measurement. This is equal to assuming $\phi = 1$. In this case,

$$\omega_i = \frac{2}{2n} = \frac{1}{n}. \tag{6.116}$$

This is in line with our prior derivation in (3.108).

In summary, it was our goal to model measurements with autocorrelated errors within the optimization problem. After deriving the equations for AR(1) NLLSQ, we noted that this leads to two drawbacks. First, our usual representation of the optimization problem as a pose graph does not work anymore. This is due to additional interdependencies between the state variables induced by the autocorrelation, which cannot be modeled with conventional pose graph elements. Second, these additional interdependencies decrease the sparsity of the problem. This leads to an increased solution time and is thus unfavorable.

As a next step, we have therefore derived new graph elements to represent constraints with autocorrelated noise. We did this for both odometry edges and observed nodes. To sum it up, for each such constraint we need to adapt the involved covariance matrices and additionally add either a hyperedge $\mathrm{z}_m^{\mathrm{w},\mathrm{AR}}$ or $\mathrm{z}_n^{\mathrm{v},\mathrm{AR}}$. This depends on whether we are handling an odometry edge or an observed node. This is a nice result as it gives us a way of reasoning about constraints with autocorrelated noise within our usual graph representation. Still, however, we suffered from the decreased sparsity.

To tackle this challenge, we have shown that we can map AR(1) NLLSQ onto ordinary NLLSQ by scaling the information matrices of all constraints with autocorrelated noise. We derived a closed-form solution for the scaling factor. Applying it is as easy as simply multiplying the corresponding information matrices with it. This makes it easy and fast to implement. Also, we did not have to alter the problem structure of ordinary NLLSQ at all. This means that this method leads to solving AR(1) NLLSQ problems at the same speed as an ordinary NLLSQ problem.

7. Evaluations

The experimental section is designed to support our claims made throughout this thesis. These include those made about the core estimator, the preprocessing sublayer, and the entire pose fusion layer. We conduct experiments on data gathered on real prototype vehicles between 2015 and 2017 and on simulated data to prove them. The prototype vehicles and their sensors are presented in Section 7.1. The pose sources for our experiments are introduced in Section 7.2.

We start the experimental evaluation in Section 7.3 by analyzing the core estimator. In this stage we neglect the preprocessing sublayer by not using any of its modules. In Section 7.4 we investigate exclusively the preprocessing sublayer. Finally, we plug them both together and take a closer look at the pose fusion layer in Section 7.5.

7.1. Vehicles and sensors

This section introduces the two prototype vehicles which were used for the development and evaluations of this thesis. Furthermore, we present the available sensors and localization systems. The sensor setups were designed by Volkswagen Group Research with two key questions in mind. First, what level of automation can we achieve with a sensor setup that is close to the current one? Secondly, what kind of sensor setup do we need for full automation?

The first vehicle is an Audi A6 Avant, whose basic architecture and sensor setup are shown in Figure 7.1. Some sensors are omitted, as these are not relevant in the context of this thesis. These include radars and a front-facing camera. Also, other components such as additional Personal Computers (PCs) or network hardware are omitted. The sensor setup is close to those of modern upper-class vehicles. The main use case for this vehicle is to drive automated on highways and rural roads.

characteristic	value
sensor type	GNSS receiver
channel count	14
GNSS capabilities	GPS, GLONASS
signal tracking	GPS L1
horizontal position accuracy	single point L1 $1.5\,\mathrm{m}$ root mean square (RMS)
data rate	$5\,\mathrm{Hz}$ (up to $10\,\mathrm{Hz}$)

Table 7.1.: Technical specification of the OEMStar receiver.

The second vehicle is a Volkswagen e-Golf 7. The basic architecture and sensor setup are shown in Figure 7.2, and a photo is shown in Figure 7.3. Again, several components are omitted for the sake of clarity. The e-Golf is all in all a more recent prototype compared to the Audi A6. This results in more recent PCs and a refined sensor setup. As this car drives fully automated in urban environments, additional lidar scanners with a higher resolution are necessary to observe the surrounding traffic participants and objects.

7.1.1. GPS receivers

Both prototype vehicles are equipped with GPS receivers. These are off-the-shelf sensors suited for automotive applications.

NovAtel FlexPak-G2 OEMStar

The NovAtel FlexPak-G2 OEMStar (OEMStar) receiver is the GPS receiver of the Audi A6. Its technical specifications are given by NovAtel Inc. (2015). Important aspects are detailed in Table 7.1.

u-blox M8 ADR

The u-blox M8 ADR (u-blox) is a GNSS receiver with an integrated IMU. Its specifications are given by u-blox (2017). Important aspects are detailed in Table 7.2.

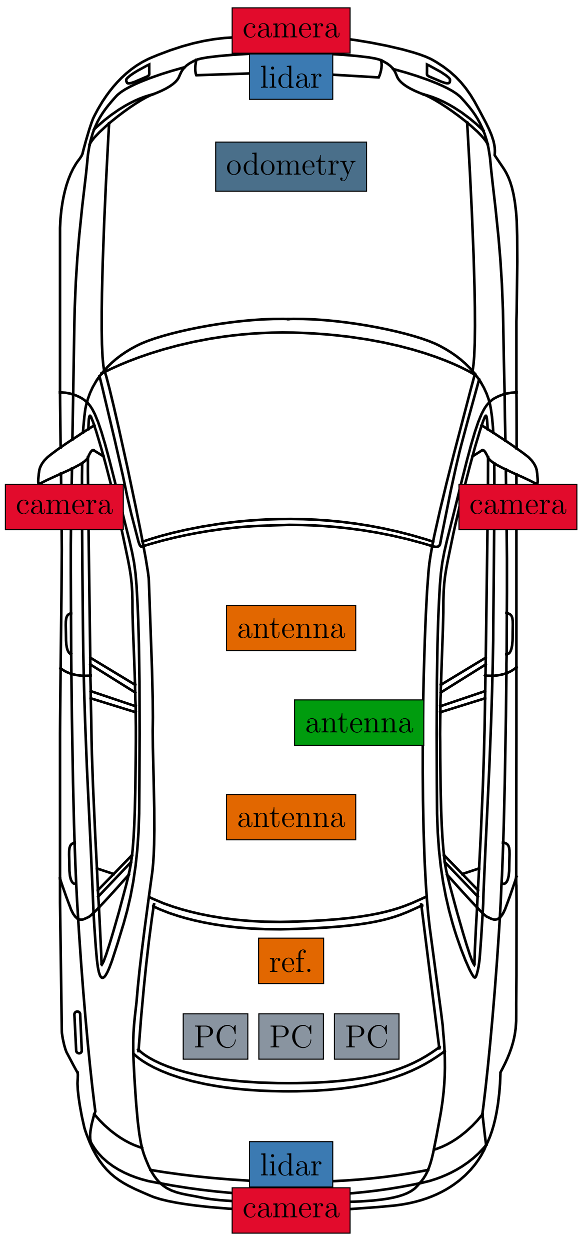


Figure 7.1.: Basic architecture and sensor setup of the Audi A6 prototype vehicle. The lower-cost GPS receiver whose antenna is represented in green is detailed in Section 7.1.1. The reference system is the Applanix system presented in Section 7.1.2. Its antennas are shown in orange together with the corresponding processing unit. The lidar scanners are the low-resolution scanners described in Section 7.1.3. The cameras are detailed in Section 7.1.4. The odometry module is presented in Section 7.2.1.

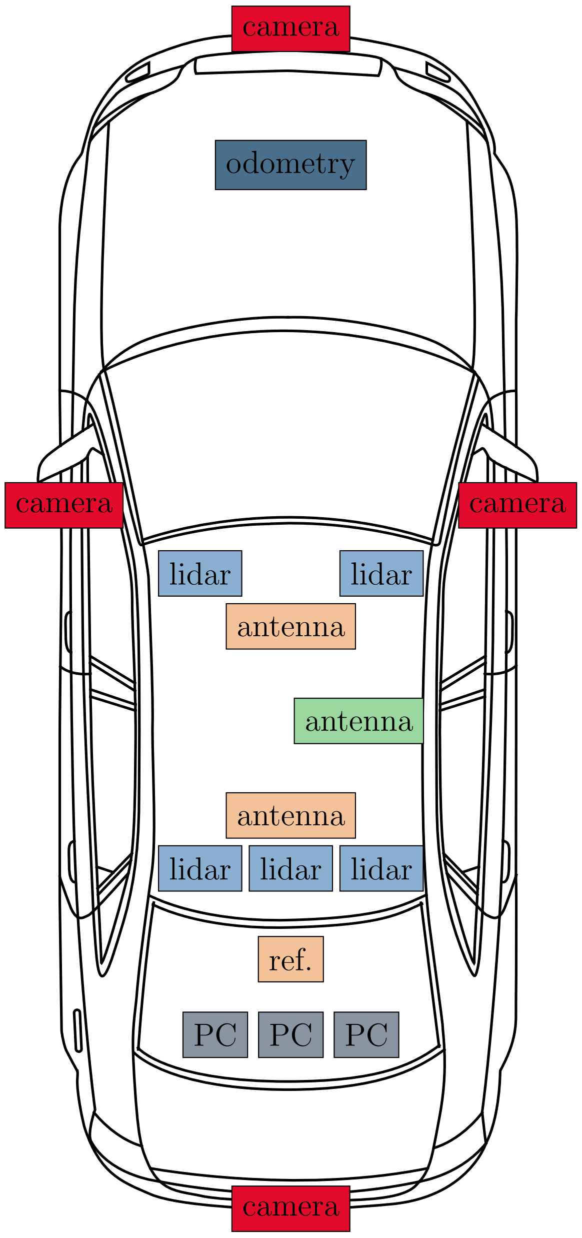


Figure 7.2.: Basic architecture and sensor setup of the Volkswagen e-Golf prototype vehicle. The slightly different color scheme indicates that the same type of sensors as in the Audi A6 are used, but different models. The lower-cost GPS receiver whose antenna is represented in green is detailed in Section 7.1.1. The reference system is the Applanix system presented in Section 7.1.2. Its antennas are shown in orange together with the corresponding processing unit. The lidar scanners are described in Section 7.1.3. The cameras are detailed in Section 7.1.4. The odometry module is presented in Section 7.2.1.

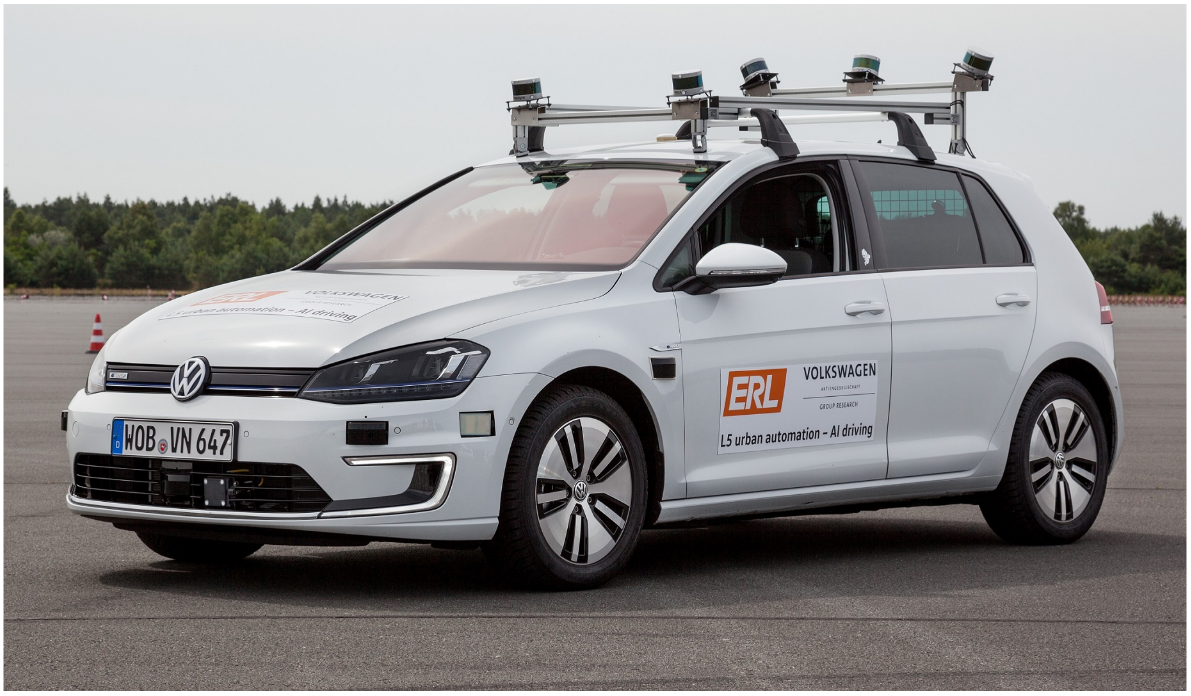


Figure 7.3.: An e-Golf prototype vehicle. Source: Volkswagen AG (2017).

characteristic	value
sensor type	GNSS receiver with integrated IMU
channel count	72
GNSS capabilities	GPS, Quasi-Zenith Satellite System (QZSS), GLONASS, Galileo, BeiDou
signal tracking	GPS L1 C/A
horizontal position accuracy	2.0 m circular error probable (CEP)
data rate	1 Hz (up to 20 Hz)

Table 7.2.: Technical specification of the u-blox receiver.

characteristic	value
sensor type	RTK-GPS receiver with integrated IMU and postprocessing
GNSS capabilities	GPS, QZSS,
	GLONASS, Galileo, BeiDou
accuracy	X, Y position $0.02\,\mathrm{m}$ RMS
	Z position $0.05\,\mathrm{m}$ RMS
	roll and pitch $0.005°$
	true heading $0.015°$
data rate	$200\,\mathrm{Hz}$

Table 7.3.: Technical specification of the Applanix POS LV 510.

7.1.2. Reference localization systems

The prototype vehicles are equipped with reference localization systems. Their main purpose is to evaluate the performance of the developed localization systems.

Applanix POS LV 510

The Applanix POS LV 510 is a GNSS receiver with an integrated IMU. The technical specifications are given by Applanix (2017). Important aspects are detailed in Table 7.3.

Oxford OXTS RT3003

The Oxford OXTS RT3003 is a GNSS receiver with an integrated IMU. The technical specifications are given by Oxford Technical Solutions Ltd. (2016). Important aspects are detailed in Table 7.4.

7.1.3. Lidar scanners

A lidar scanner serves to generate three-dimensional point clouds of the environment. It measures the distance and reflectivity of the closest objects for a discrete set of horizontal and vertical angles. To this end, it emits laser pulses and measures the time-of-flight until the laser pulse is reflected and returned to the sensor in the scanner. Conventionally, they use rotating mirrors to control the horizontal angle of the emitted laser pulses.

characteristic	value
sensor type	RTK-GPS receiver with IMU
GNSS capabilities	GPS, QZSS,
	GLONASS, Galileo, BeiDou
accuracy	X, Y, Z position $0.01\,\mathrm{m}$ CEP
	velocity $0.05\,\frac{\mathrm{km}}{\mathrm{h}}$
	roll and pitch $0.03°$
	heading $0.1°$
data rate	$100\,\mathrm{Hz}$

Table 7.4.: Technical specification of the Oxford OXTS RT3003.

Newer generations try to avoid such moving mechanical components to gain a higher durability.

Two different types of lidar scanners are deployed on the prototype vehicles. The Audi A6 uses a front- and a rear-facing Valeo ScaLa scanner, which is described in Section 7.1.3. The e-Golf features five Velodyne VLP-16 Puck, which is described in Section 7.1.3.

Valeo ScaLa

The Valeo ScaLa is an automotive-grade lidar scanner. It uses three physical scan layers with a horizontal angular range of $145°$. After each full sweep, it tilts its internal mirror to cover an additional scan layer during the next sweep. After that, it tilts back into its initial position. Table 7.5 lists the main characteristics of the scanner.

Velodyne VLP-16 Puck

The Velodyne VLP-16 Puck is a lidar scanner with a horizontal angular range of $360°$, thus providing a full horizontal scan. Its main characteristics are listed in Table 7.6. Further details can be found in the official documentation (Velodyne LiDAR Inc., 2017).

characteristic	value
measurement principle	laser scanner based on time-of-flight measurements
scan rate	25 scans per second, 750 rpm, 581 laser pulses per scan
scan range	$150\,\mathrm{m}$ nominal, $100\,\mathrm{m}$ effective
wavelength of sender	$905 \pm 10\,\mathrm{nm}$
number of scan layers	three physical layers that tilt vertically, resulting in four effective layers
number of scan points per second	$43\,575$
horizontal scan rate	$145°$
horizontal angle resolution	$0.25°$
vertical scan range	$2.6°$ to $3.6°$
vertical angle resolution	$0.8°$
distance resolution	$100\,\mathrm{mm}$

Table 7.5.: Technical specification of the Valeo ScaLa scanner.

characteristic	value
measurement principle	laser scanner based on time-of-flight measurements
rotational speed	5 to 20 rotations per second (adjustable)
scan range	$100\,\mathrm{m}$ maximum
wavelength of sender	$903 \pm 7\,\mathrm{nm}$
number of scan layers	16 channels
number of scan points per second	up to $288\,000$
horizontal field of view	$360°$
horizontal angle resolution	$0.1°$ to $0.4°$, depends on the rotational speed
vertical field of view	$30°$ $(+15°$ to $-15°)$
vertical angle resolution	$2°$
distance resolution	$\pm\,3\,\mathrm{cm}$

Table 7.6.: Technical specification of the Velodyne VLP-16 Puck.

characteristic	value
sensor type	monocular fisheye cameras
placement	front, left, right, and rear side of the vehicle
frames per second	30
image resolution	1280×960 pixel
color depth	16 bit
opening angle horizontal	190°
opening angle vertical	123.8°

Table 7.7.: Technical specification of the top view cameras.

7.1.4. Top view cameras

Both prototype vehicles feature four top view cameras. They are installed in the right and left side mirrors as well as above the front and rear license plate. These fisheye cameras use a rolling shutter to acquire images. The images are used for detecting lane markings. Table 7.7 lists the main characteristics of the cameras.

7.2. Self-localization systems

This section presents the self-localization systems that are available in at least one of the prototype vehicles presented in Chapter 7.1. They serve as input sources for the pose fusion. Overall, there are two local and five global pose sources available. We describe these in the following sections.

7.2.1. EgoMaster

The EgoMaster is a software module that computes the current egomotion of the vehicle. The egomotion is defined as the motion of the vehicle within its environment, whereas the scene itself is considered to be static. This estimation is performed in the vehicle reference frame as described in Section 3.1.1. It relies on sensor information from IMUs, angular rate, steering angle, wheel speed, and chassis lift sensors. All of these sensor readings are readily available as state-of-the-art controller area network bus (CAN bus) messages from the ESP and anti-lock braking system (ABS) sensors. The CAN bus is a vehicle bus over which electronic control units (ECUs) exchange data. To provide a consistent estimate of the egomotion, the EgoMaster fuses the sensor readings with the help of an EKF. An early version of this filter is described by Baer et al. (2009). In summary, the EgoMaster can be regarded as a classical odometer. It is available for both the Audi A6 and the e-Golf and provides local poses with $100\,\mathrm{Hz}$.

7.2.2. Landmark-based localization

The landmark-based localization (LBL) is a software module that computes the map-relative position of the vehicle based on comparing its local landmark detections to a global landmark map. Landmarks are static and persistent objects that have a semantic meaning. These include traffic signs, traffic lights, lane markings, building faces, curbs, or vertical pole-like objects. They are therefore different to visual features like the popular scale-invariant feature transform (SIFT) features, which are often used for visual SLAM algorithms. In addition, they are not bound to a certain sensor type as they can potentially be detected by multiple sensors, such as cameras, lidars, or radars.

For the sake of a simple explanation we pick an example of landmark detections to illustrate the general approach. Figure 7.4 shows the detection of lane markings in camera images. For this purpose, the camera images from the top view cameras as described in Section 7.1.4 have been undistorted and transformed to a virtual top view image. Subsequently, the background of the image is subtracted and contours are found after binarizing the image. These contours represent the landmark candidates. A classifier learns in a supervised learning step the classification of these candidates into dashed or solid lane markings, arrows, stop lines, and unknown objects. The visualization shows the landmark classifications in the virtual top view image.

With the help of other detectors we are able to find a range of different landmarks within close proximity of the vehicle. These are tracked in a subsequent landmark tracking step. To estimate the vehicle's pose, we compare this set of detections to a previously constructed landmark map. Figure 7.5 depicts an example of a landmark map, that contains dashed and solid lane markings as well as poles. As the landmark maps are globally referenced, we obtain both a map-relative and a global pose. Finally, the pose is filtered over time with a particle filter. A more detailed description of the LBL is given by Stess (2017). In total, this system provides global poses at a nominal frequency of $5\,\mathrm{Hz}$.

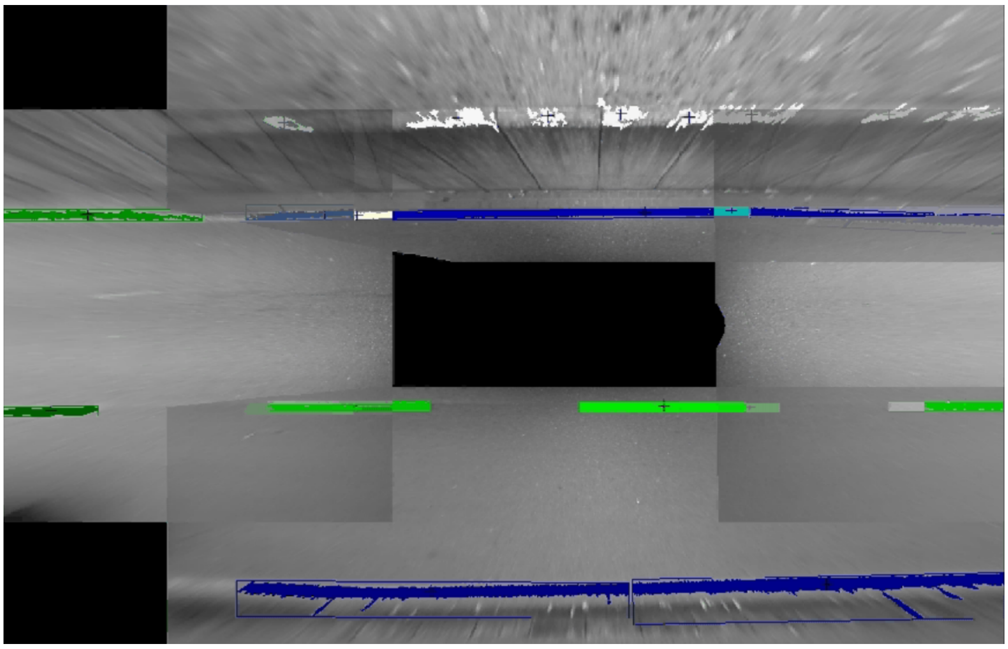


Figure 7.4.: Detection and classification of lane markings in top view camera images. Green are dashed line markings, while blue boxes visualize solid road markings.

7.2.3. Lidar localization

The lidar localization (LiLoc) is a software module that matches lidar scans against a globally referenced point cloud. It makes use of and is tailored to the lidar scanners described in Section 7.1.3. As these are only available on the Audi A6, this global pose source is not available for the e-Golf. This module provides global poses with a nominal frequency of $2\,\mathrm{Hz}$.

7.2.4. Localization sensor data fusion

The localization sensor data fusion (LSDF) is a software module that computes both a global and a local pose. It is based on an error state EKF that provides a tight coupling between GNSS and IMU data. We refer to its global pose output as *LSDF*, and to its local pose output as *LSDF relative*. The system is only available on the Audi A6 due to the close integration with its IMU. Scheide et al. (2011) describes the overall system and its architecture in more detail. The nominal frequency of both outputs is $50\,\mathrm{Hz}$.

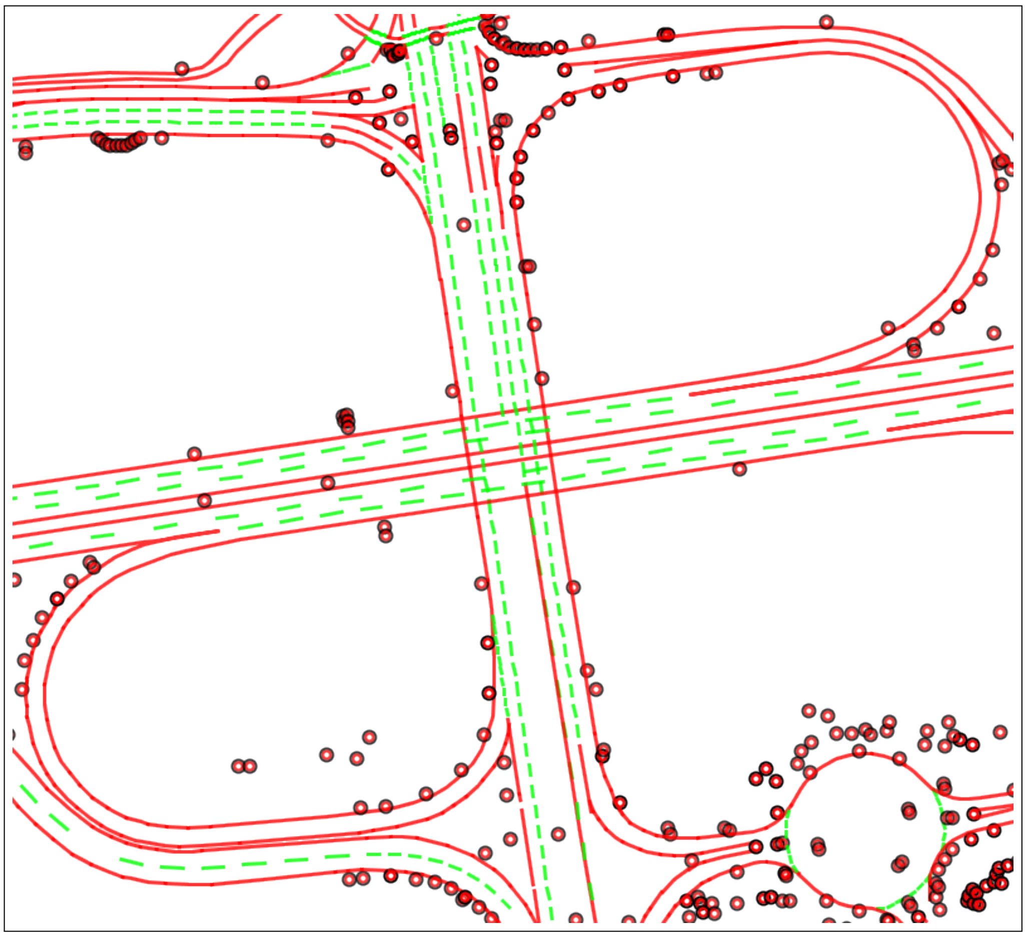


Figure 7.5.: A landmark map. It contains dashed lane markings (green), solid lane markings (red), and poles (black circles).

7.2.5. GPS receivers

We present in Section 7.1.1 the available GPS receivers on the prototype vehicles. For the sake of completeness, we briefly list them here again: on the Audi A6, there is an OEMStar receiver available (see Section 7.1.1). On the e-Golf, there is an u-blox receiver installed (see Section 7.1.1). These two receivers provide global poses to the pose fusion with $5\,\mathrm{Hz}$ and $1\,\mathrm{Hz}$ respectively.

7.3. Core estimator

The experiments in this section serve to investigate the core estimator presented in Section 5 while blending out the preprocessing modules. They are based on simulated data to emphasize some key properties of the core estimator. We presents experiments with real data and the core estimator in Section 7.5.

The ground truth was obtained by postprocessing the recordings of the Applanix system presented in Section 7.1.2. We recorded the data during a drive of about $16\,\mathrm{km}$ in rural and urban areas in Germany. By adding artificial noise to this ground truth trajectory we generate the input data for our core estimator. The advantage of creating simulated data in this way over purely generating ground truth and noisy data is that this approach inherently leads to a natural and realistic trajectory.

All input sources are sampled around the true values from a Gaussian distribution with a standard deviation of $3.0\,\mathrm{m}$ in both lateral and longitudinal direction and $4°$ heading orientation of the vehicle. Figure 7.6 illustrates a small-scale application of the core estimator. It is a screenshot of the visualization of the PGF, which is our implementation of the core estimator. This example illustrates how a small chain pose graph with ten hidden nodes and two simulated input sources looks like. Also, we can see the prior node at the end of the graph in purple. These graphs, albeit in larger size and with different parameters, form the basis for our experiments in this section.

We divide the core estimator experiments in four categories. The first category is outlined in Section 7.3.1 and its experiments investigate the optimization of the underlying NLLSQ problem. The experiments in the second category analyze the effect of multiple input sources. In the third category we treat the question of how the number of hidden nodes influences the estimation quality. Lastly, we examine the runtime requirements

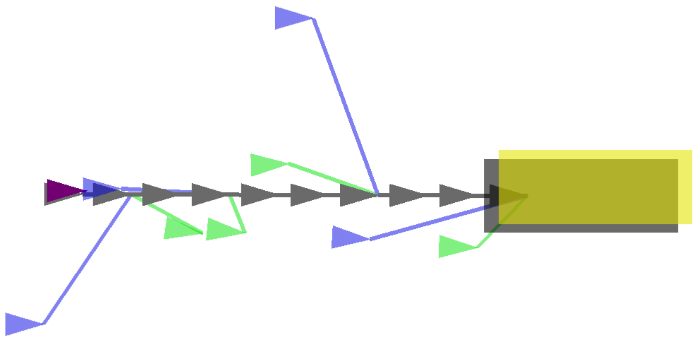


Figure 7.6.: Illustration of the core estimator's pose graph in the experiments. The gray nodes visualize its hidden nodes, the green and blue nodes are observed nodes stemming from global pose measurements. The purple node is the prior node. The gray rectangle represents the vehicle pose as computed by the core estimator, whereas the yellow rectangle represents the ground truth vehicle pose.

of the PGF.

7.3.1. Optimization

Fundamentally, the core estimator is solving the optimization problem of finding those hidden nodes which best explain all pose constraints. We outline in Section 3.4.2 two common algorithms to solve this optimization problem, the Gauss-Newton and the Levenberg-Marquardt algorithm. Both are suitable for our application. Therefore, we start off the experimental section by examining the differences of these two algorithms for our use case and with our data. Ideally, we are able to identify one of the two as the most adequate for our core estimator. We assess them by asking how many iterations do both algorithms need until they converge? Following up on that question we ask whether they converge to the same solution.

We answer both questions by conducting an experiment based on ground truth data from an Applanix trajectory and generate observations from eight global and four local pose sources. We set the maximum number of hidden nodes to $M = 300$ and the temporal resolution to $\varDelta t = 25\,\mathrm{ms}$. With the output frequency set to $f = \frac{1}{150\,\mathrm{ms}}$, the PGF produces a total of roughly 4600 output pose estimates per single run of an experiment. For both Gauss-Newton and Levenberg-Marquardt, we do the following:

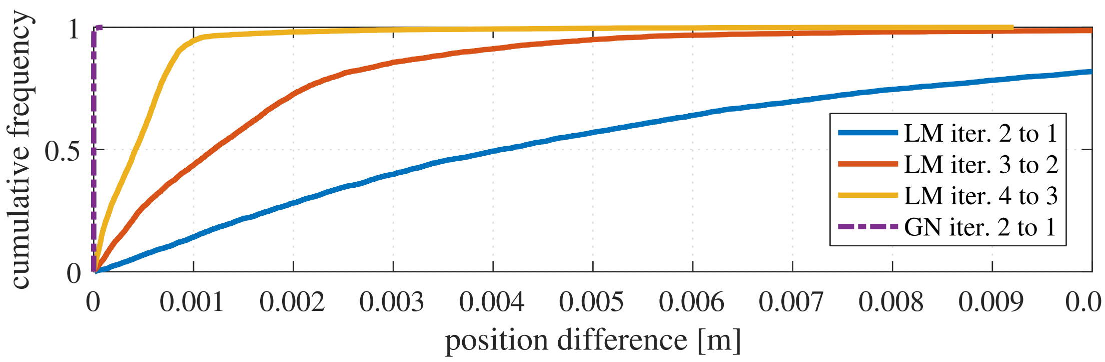


Figure 7.7.: Empirical distribution function of the position difference of the output pose estimates of the core estimator when using Gauss-Newton (GN) and Levenberg-Marquardt (LM) with different numbers of optimization iterations.

- Initially, we set the number of optimization iterations to $i = 1$.
- We keep this number fixed and process all simulated data within the PGF, virtually driving the trajectory of $16\,\mathrm{km}$. This produces each time the 4600 output pose estimates mentioned above.
- Subsequently, we increase i by one and repeat the experiment until the estimation result does not change anymore. Our criterion for convergence is that none of the output pose estimates changes by more than $1\,\mathrm{cm}$ compared to the previous run.

For the evaluation of this experiment, we plot the empirical distribution function (EDF) of the Euclidean position difference of all 4600 output pose estimates for a given number of optimization iterations i. Figure 7.7 displays this plot, showing for both optimization algorithms and for different values of i the cumulative frequency of the position differences of the output pose estimates. The position differences are obtained by computing the Euclidean distance between all output pose estimates at the same timestamp during the different runs of the experiment. We observe that the output of the core estimator does not change significantly anymore after setting the number of optimization iterations $i = 3$ for Levenberg-Marquardt. The maximum position difference in this setting falls to $0.92\,\mathrm{cm}$. Using Gauss-Newton we observe that a single optimization iteration suffices. The maximum position difference after the second optimization iteration is already at roughly $0.01\,\mathrm{cm}$.

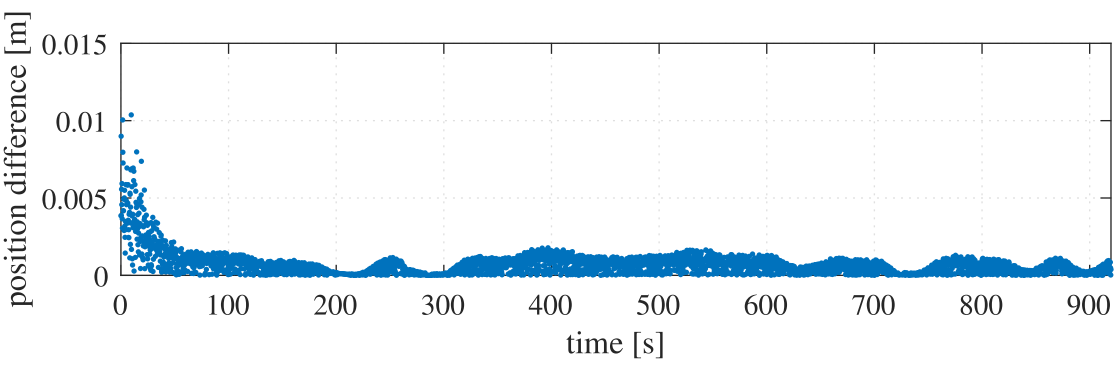


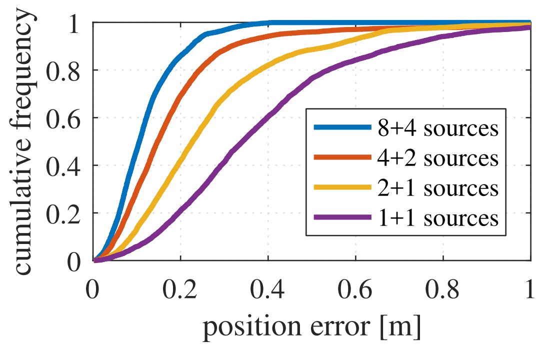
Figure 7.8.: Position difference of the output pose estimates for Gauss-Newton and Levenberg-Marquardt over time. The difference is negligible. Therefore, both algorithms converge to the same solution.

We attribute this high convergence speed to several factors. First, our fusion problem is usually well constrained with a global pose measurement every few hidden nodes. In contrast, the optimization of pose graphs that almost purely consist of odometry constraints with a few loop closures generally takes more iterations to converge[1]. Secondly, our chain pose graphs by design prevent any edges between non-consecutive nodes, making the structure less complex. Thirdly, our sliding window approach provides the optimization result of the last time step as the initial guess for the current time step. As most of the observed and hidden nodes within the sliding window stay identical, this leads to a very good initial guess for the optimization. Lastly, the graph construction method of the PGF builds up the graph by setting intelligent guesses for the initial poses of hidden nodes by considering the local and global constraints. This results in good initial guesses for hidden nodes that are added to the sliding window.

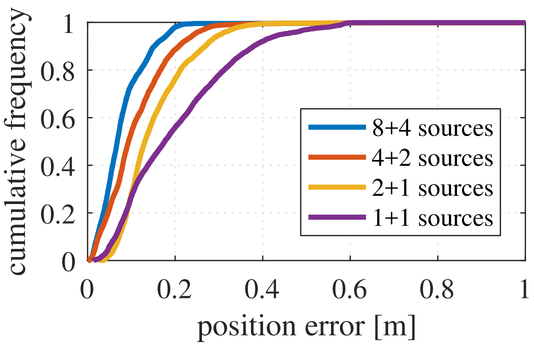
After finding that Gauss-Newton converges after a single optimization iteration and that Levenberg-Marquardt needs three, we would like to know whether both algorithms converge to the same solution. We therefore compare the obtained output pose estimates for Gauss-Newton and Levenberg-Marquardt after convergence. Figure 7.8 shows their position differences in each time step. As we can see, the position difference falls from about $1\,\mathrm{cm}$ in the initialization phase to less than $0.02\,\mathrm{cm}$ during normal operation mode. We conclude that both algorithms lead to the same results.

In summary, we have experimentally determined that Gauss-Newton takes a single

[1]See the experiments in Section 2.5.2 by Kümmerle (2013) for examples.



(a) Online fusion without a prior node.



(b) Online fusion with a prior node.

Figure 7.9.: EDF of the position error of the core estimator for different number of input sources for online fusion. The legend entries *"$a + b$ sources"* denote a global and b local pose sources. The position quality increases for an increasing number of input sources. Enabling the prior node provides a temporal filtering which further reduces the position error.

optimization iteration to converge and that Levenberg-Marquardt takes three. Moreover, both algorithms converge to the same solution. Given these properties, we choose Gauss-Newton as our default optimization algorithm because needing less optimization iterations means needing less computation time. This gives us the flexibility to increase the output frequency of the PGF, to increase the number of hidden nodes, or to process more input sources.

7.3.2. Number of input sources

The second experiment with simulated data is designed to show that the position error decreases for an increasing number of input sources. To this end, we run the PGF repeatedly on the test dataset and increase the number of input sources each time. As an additional parameter we enable and disable the prior node to examine its influence.

Figure 7.9 depicts the EDF of the position error of the core estimator. It shows how the fusion result improves for an increasing number of input sources. This is true regardless of whether the prior node is enabled. However, we can see a clear increase in position quality when using the prior node. It provides a temporal smoothing that further reduces the error.

With the same dataset we also examine the fusion result of the offline batch opti-

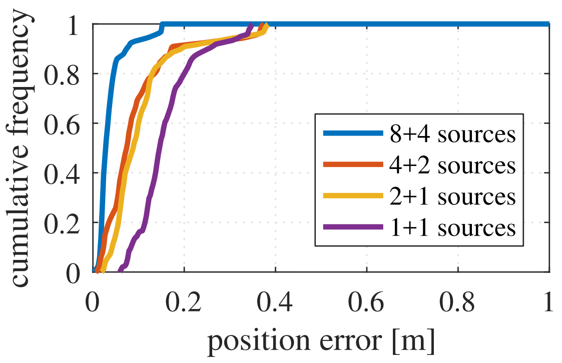


Figure 7.10.: EDF of the position error of the core estimator for different number of input sources for offline fusion. The position error of the offline batch optimization decreases for an increasing number of input sources.

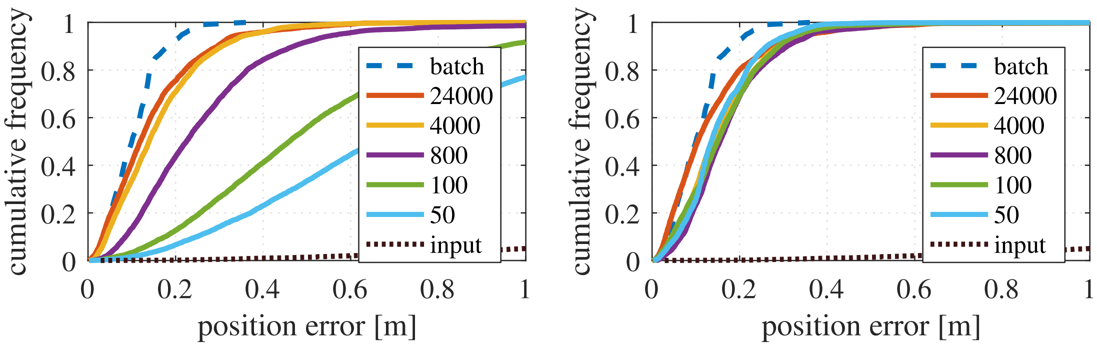
mization. This means that we build up the entire chain pose graph over the dataset and optimize it at the very end. We again perform this for different number of input sources. Figure 7.10 shows the EDF for the position errors. The small vertical jumps in the curves are caused by standstill phases during the drive, for example in front of traffic lights. All output fusion estimates during these phases lead to the same position error, which in turn leads to the vertical jumps in the cumulative frequency. Similarly to the online fusion, we observe that also the offline batch optimizations profit from more input sources.

In summary, an increasing number of input sources reduces the position error of the core estimator. This is true for both the online and offline fusion. Also, using a prior node further reduces the error during online operation.

7.3.3. Number of hidden nodes

The next experiment on simulated input data serves to investigate the estimation quality as a function of the number of hidden nodes M. This engages the question of the required number of hidden nodes and demonstrates that the sliding window estimate converges to the online batch solution for an increasing number of hidden nodes.

Similarly to the previous experiments, we evaluate the position error of the output pose estimates of the core estimator for a test dataset. We repeat this experiment and increase M in each iteration. Again, we perform this experiment with temporal filtering enabled and disabled. In this experiment, we simulate two global and one local pose



(a) Online fusion without a prior node. (b) Online fusion with a prior node.

Figure 7.11.: EDF of the position error for different numbers of hidden nodes M.

source.

Figure 7.11a shows the cumulative position error distribution for the different settings. The dotted black curve represents one of the two global pose sources, which performs poorly. The other pose source performs by construction comparatively. Using the core estimator with any of the proposed settings leads to a significant improvement. Moreover, the pose fusion results improve for more hidden nodes and approach the batch optimization quality. The position error decreases only slightly after $M = 4000$. Therefore, increasing M further yields only a disproportionate small gain in position quality.

Figure 7.11b shows the result of the same experiment when temporal filtering in the form of the prior node is enabled. This time we conclude that the variable M is not decisive for the position error. All curves lie within a small band and are virtually indistinguishable. Only the results of the setting with $M = 24\,000$ slightly outperform the others. Therefore, it is safe to use a smaller sliding window if temporal filtering is enabled. We also conclude that if we are using a sufficiently large sliding window, then it does not make a significant difference in estimation quality whether we enable or disable temporal filtering.

In summary, we illustrated our claim that the core estimator approaches the batch optimization performance for increasing sizes of the sliding window. We have also shown that the estimation quality does not depend on M if we enable the prior node. Moreover, using a large enough sliding window is equal in terms of estimation quality to using a prior node.

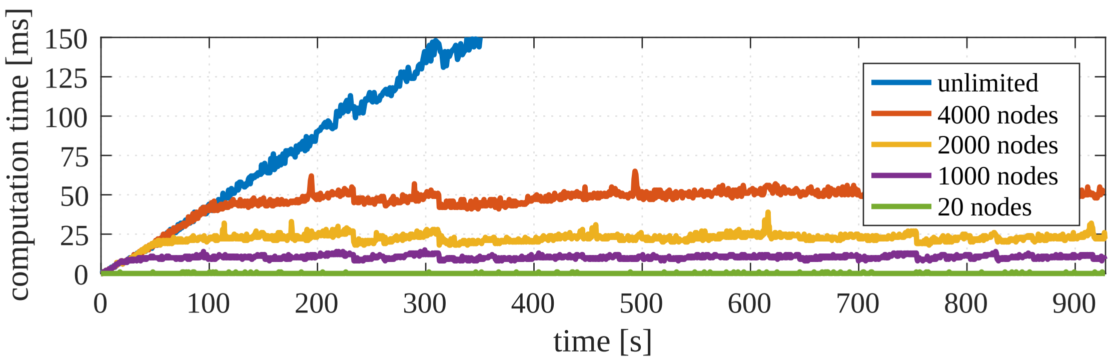


Figure 7.12.: Computation time for different numbers of hidden nodes. The blue curve corresponds to a setup in which the number of hidden nodes is unlimited such that nodes never get marginalized out. The resulting unbounded demand for computation time disqualifies it for online usage. The computation time is roughly constant in all other configurations.

Now that we have an idea about the preferable size of the sliding window, we investigate whether we can actually achieve this in an online system.

7.3.4. Runtime performance

We concern ourselves with the runtime performance of the PGF. Our goal is twofold: we show that we can solve graphs of relevant sizes online and that the optimization of chain pose graphs has a runtime complexity of $\mathcal{O}(n)$. For these purposes we conduct an experiment during which we measure how long it takes to optimize chain pose graphs of varying size.

The software was repeatedly run on a single core of a laptop with an Intel i7-4800QM processor. Figure 7.12 shows the computation time at each time step for different numbers of hidden nodes in the sliding window pose graph. The blue curve illustrates the need for limiting the size of the graph as otherwise the computation time grows unboundedly.

The near-constant computation time once the graph attains its full size is expected as the optimization of a chain pose graph of fixed size is constant. Also, the linear increase in computation time before the number of nodes equals M is in line with our theoretical expectations as the optimization of a chain pose graph has a runtime complexity of $\mathcal{O}(n)$. An empirical analysis of the data depicted as the blue curve in Figure 7.12 (unlim-

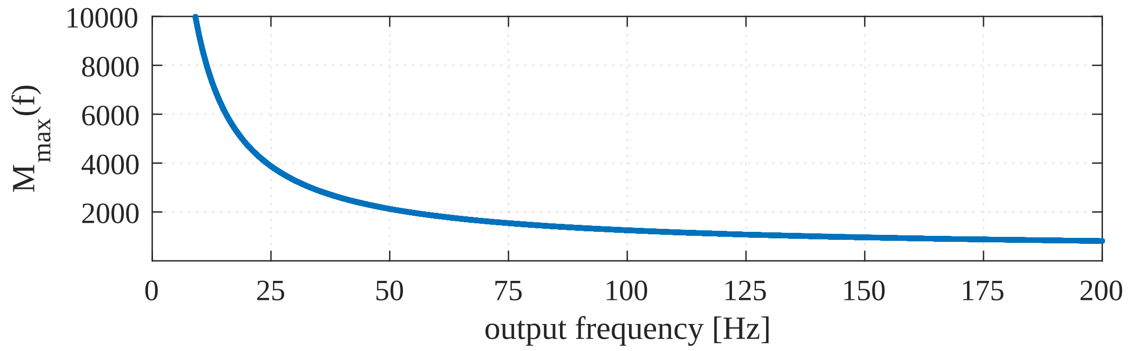


Figure 7.13.: Maximum number of hidden nodes $M_{\max}$ that can still be processed fast enough to attain the corresponding output frequency f.

ited number of hidden nodes) reveals the reciprocal relationship between the maximum number of hidden nodes $M_{\max}$ and the attainable output frequency f. We obtain this relationship by computing the maximum output frequency for each data point and subsequently fitting a curve $M_{\max}(f) = \alpha_1 \frac{1}{f} + \alpha_0$ to it. We find the coefficients $\alpha_1 = 87.41$ and $\alpha_0 = 374.32$. Figure 7.13 shows the corresponding graph.

Note that the time needed for the optimization is not directly equal to the overall computation time per cycle. Other operations, such as marginalizing old and appending new constraints, also take time. Their time needed depends on different parameters, such as frequency and number of input sources. Each constraint that we add to the graph results in at least one additional addend in $\boldsymbol{H}$ and $\boldsymbol{b}$. However, the optimization itself primarily depends on M and not on the number of constraints within the graph.

Different parametrizations allow us to balance the need for a high output frequency versus the desire for more hidden nodes. Their relationship serves as a basis for choosing f and $M_{\max}$. We investigate in the next experiment how to react at runtime to situations in which less CPU time than usual is available, i.e., situations in which the relationship depicted in Figure 7.13 temporarily does not hold.

7.3.5. Resource-adaptive state estimation

This experiment is designed to show that the PGF effectively adapts online to the available computational resources. To this end, we conduct an experiment and inspect the dynamic adaptation of the number of hidden nodes M_t. The goal is to limit the computation time c_t such that it is on average less or equal to $\frac{1}{f} = 50\,\mathrm{ms}$. To clearly see the impact of the PID controller, we deliberately maximize the CPU load multiple times during the experiment. As a consequence, the computation time increases in these phases because less CPU time is available for the PGF. The PID controller counteracts the exceeding computation time by regulating M_t. We repeat the experiment with the same CPU load pattern but without the PID controller. The computation time exceeds the setpoint of $50\,\mathrm{ms}$ by roughly 60% in phases when less CPU time is available. The overall mean computation time amounts to $71\,\mathrm{ms}$ without and to $41\,\mathrm{ms}$ with the PID controller. The progression of M_t and c_t over time are shown in Figure 7.14.

We conclude that the PGF is able to meet on average a predefined computation time limit. This, in turn, is directly related to the availability of the output pose estimates.

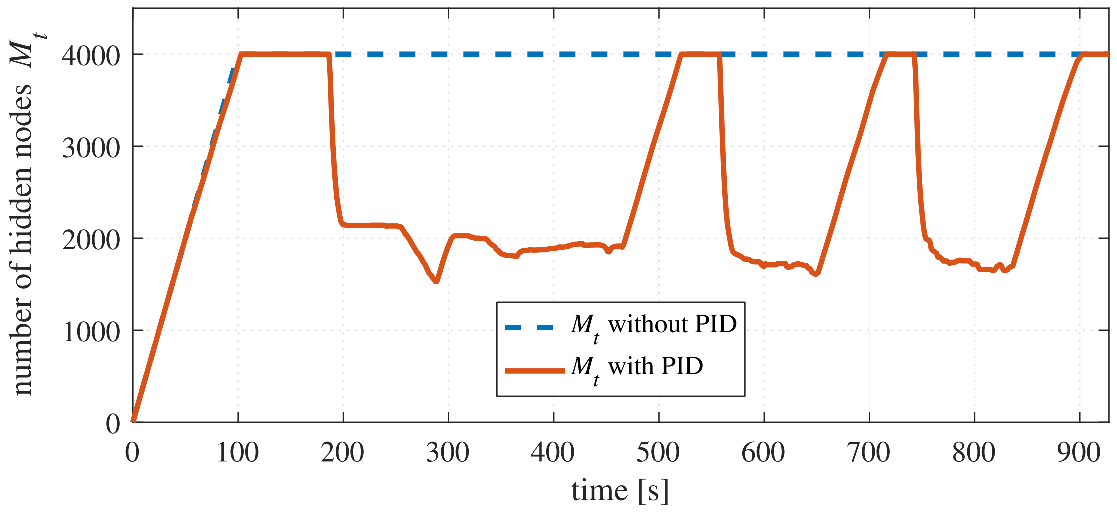
7.4. Preprocessing sublayer

In this section we evaluate the modules of the preprocessing sublayer separately. Having four modules within the preprocessing sublayer, we divide the experiments into four parts.

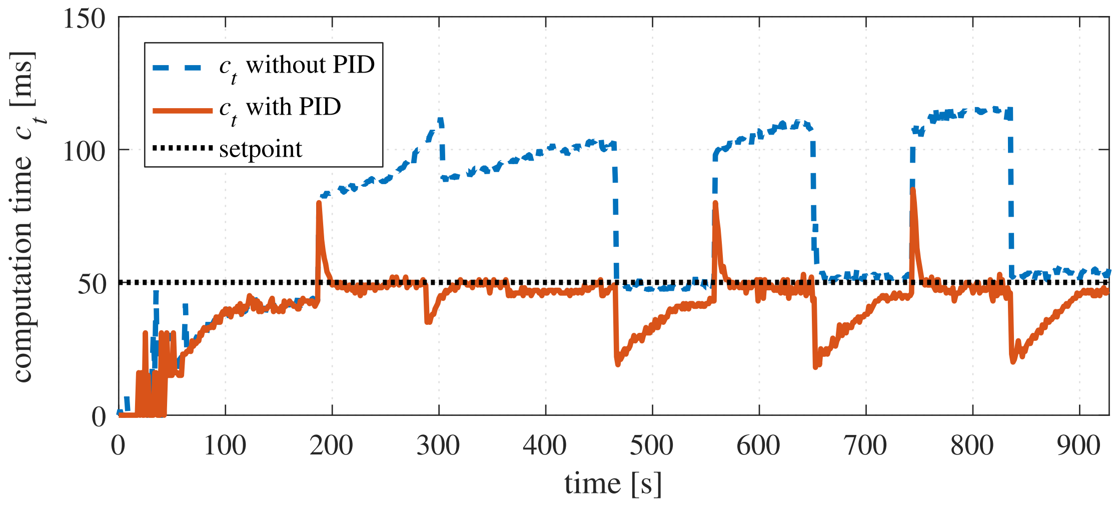
7.4.1. Bias estimation

The following experiment is designed to show that the bias estimation effectively reduces the time-varying, systematic bias of GPS data gathered on a real prototype vehicle and that the pose fusion produces precise estimates. We provide more evaluations of this module in the experiments presented in Section 7.5. The vehicle is the Audi A6 Avant presented in Section 7.1. It is equipped with two GPS receivers of different quality. We make use of the pose estimates from the OEMStar receiver as input source. The Applanix system comes into play as the reference pose source. The EgoMaster provides an estimation of the local movements of the vehicle. The data was recorded on a route of about $16\,\mathrm{km}$ in rural and urban areas in Germany.

As a first step, we need to determine the optimal length of the sliding window s^*. If it is too long, then the estimated bias adapts too slowly to the actual bias. If it is too short, then we introduce an artificial correlation between the unbiased and the biased pose source. We have given in (6.5) a way to calculate s^*. It consists in an exhaustive search



(a) The PID controller limits the number of hidden nodes M_t.



(b) The limitation of M_t results in the limitation of c_t.

Figure 7.14.: Effect of the resource-adaptive online state estimation. A PID controller dynamically controls the number of hidden nodes to attain a predefined setpoint for the computation time. We maximize the CPU load in three sequences and observe how M_t is controlled (see (a)) such that c_t stays as much as possible beneath the setpoint (see (b)).

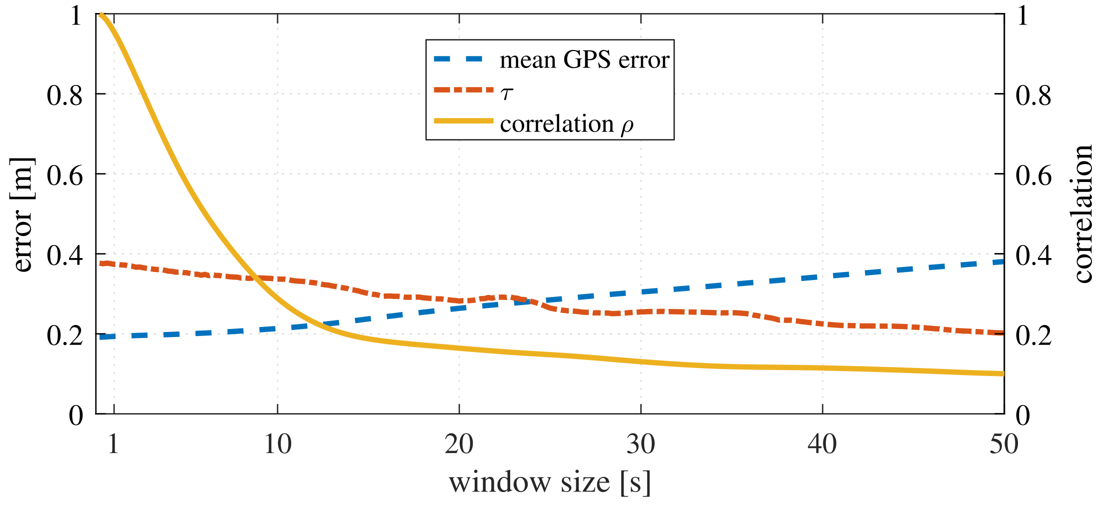


Figure 7.15.: We find the optimal sliding window size by minimizing the mean error of the biased pose source (depicted as GPS error) while limiting the correlation of its error with the error of the pose estimates of the unbiased source (see (6.6)). These two objectives conflict because the mean GPS error increases (as the adaption of the sliding window is too slow) and the correlation decreases to random correlations with larger window sizes. Additionally, we ensure that the sliding window is large enough by inspecting the maximum mean error over all sliding windows (see (6.7)).

over all windows sizes of interest. For each window size, three metrics are checked: How well is the bias estimated? Is the correlation between the unbiased and the biased pose source negligible? Is the sliding window long enough such that we can assume that the error over the unbiased pose source within the sliding window is almost zero?

Figure 7.15 shows the output of the optimization over a given dataset with pose estimates from the OEMStar receiver. The optimal sliding window size in terms of mean GPS error should be equal to one because the unbiased source should be much more precise than the GPS-based pose estimates. However, this makes the biased equal to the unbiased pose estimates and yields a maximum correlation $\rho(\boldsymbol{z}^b - \boldsymbol{p}, \boldsymbol{z}^u - \boldsymbol{p})$ of their error terms. Also, that sliding window is too small to justify the assumption of a near-zero mean of the error of the unbiased source as indicated by the value of $\tau(\boldsymbol{z}^u, \boldsymbol{p}, s)$. We therefore settle for a window size of $20\,\mathrm{s}$.

We fuse the pose estimates of the OEMStar receiver with the EgoMaster data. In this process, we estimate the bias with the help of the presented sliding window technique by

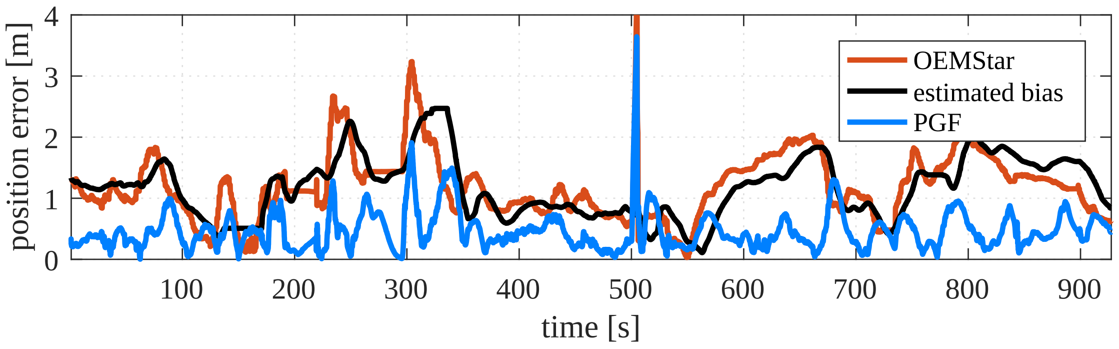


Figure 7.16.: Position error of the OEMStar receiver over time. The estimated bias is shown in black. It serves to reduce the systematic, time-varying bias of the data. The biased pose estimates (red) exhibit a higher position error than the bias-reduced pose fusion output (blue).

comparing against a third pose measurement source, which is afterwards ignored for the remainder of the pose fusion for the sake of clarity. Figure 7.16 shows the position errors over time of the OEMStar receiver, the estimated bias, and the error of the resulting output of the PGF. The graph of the estimated bias resembles the graph of the position error of the OEMStar receiver. Ideally, these two would be equal. The estimated bias seems to be slightly shifted compared to the position error of the OEMStar receiver. This is due to the usage of a sliding window. The position error of the PGF is significantly lower than the error of its input source.

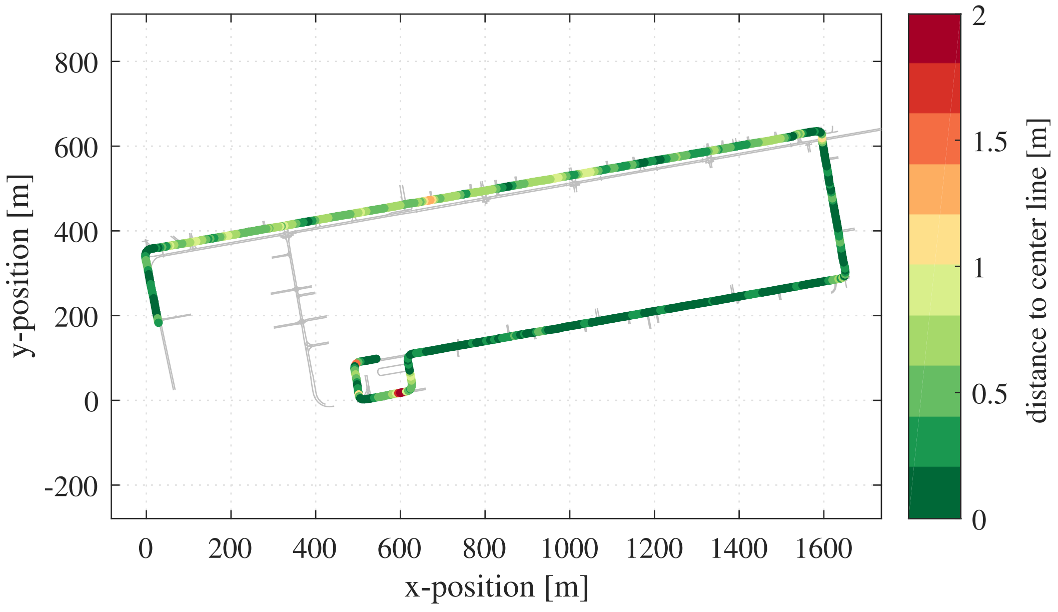
The pose estimates of the OEMStar receiver exhibit a mean absolute error (MAE) of $1.19\,\mathrm{m}$ with a standard deviation of $0.52\,\mathrm{m}$. The bias estimation reduces this by a factor of two to a MAE of $0.47\,\mathrm{m}$. The MAE of the pose fusion is in the same order of magnitude with $0.47\,\mathrm{m}$ and a standard deviation of $0.33\,\mathrm{m}$, thus showing the effectiveness of the bias estimation.

7.4.2. Map-based outlier handling

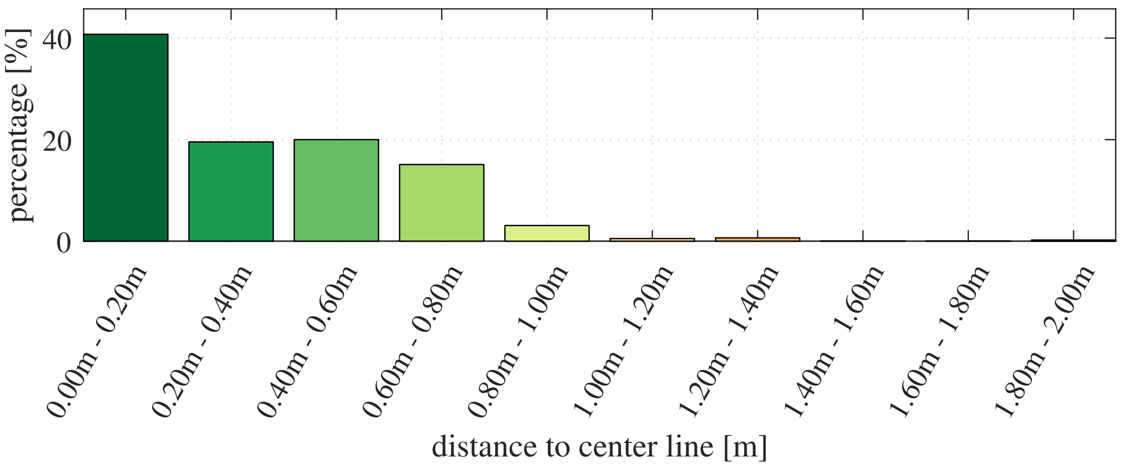
In this section, we evaluate our method of down-scaling pose estimates which are far off the road. The experiment is designed to evaluate our claims that a vehicle is usually driving close to the center lines and that down-scaling bad pose estimates increases the pose fusion performance. Moreover, we compare the method to using a Pseudo-Huber cost function. With the help of a typical situation, we make the argument that in certain situations it is favorable to employ our method over using robust cost functions. Moreover, we provide data to justify the choice of the method's design parameters.

Our method is based on the assumption that a vehicle is most of the time driving close to the center line. Therefore, it is our first step to validate this assumption. We check it by evaluating the distance of the reference system to the next center line. For this we record a challenging data set in an urban scenario. In this environment, the vehicle changes lanes, needs to go around pedestrians, has to adapt its trajectory to the traffic around it, or needs to sidestep slightly to avoid parked cars on the side. These behaviors can lead to violations of our assumption, making the data set challenging. Even in this environment we observe that the vehicle is driving most of the time close to the center line. Figure 7.17a shows the trajectory of the vehicle. The distance of the reference system to the next center line is color-coded. Large portions of the trajectory are green, indicating a low distance. Figure 7.17b supports this evaluation with a histogram of the distances. In about $40\,\%$ of the time, the vehicle is less than $20\,\mathrm{cm}$ away from the center line. This increases to about $98\,\%$ of the time in which the vehicle is closer than $1\,\mathrm{m}$. The remainder amounts to the challenging situations named above. We conclude that we can assume that the vehicle is driving most of the time less than $1\,\mathrm{m}$ away from the center line.

Using this data we determine the scaling function parameters λ_1, λ_2, $s_{\min}$, and $s_{\max}$ as introduced in (6.10) in Section 6.2. The parameter $s_{\max}$ sets the maximum value of the scaling function. As a first step we set it to $s_{\max} = 1.0$. Higher values would indicate that the scaling potentially adds information to the pose estimate, which it does not do. Lower values indicate that the scaling will downscale any kind of pose estimate, irregardless of its distance to the center line. Secondly, λ_1 defines the width of the plateau for which $s(\delta) = s_{\max}$. Based on the evaluation in Figure 7.17b, it makes sense to set this to $\lambda_1 = 1.0$. This means that we generally assume the vehicle to be in a corridor of $1\,\mathrm{m}$ around a center line. Any pose estimate within this corridor it not scaled, as it could likely be close to the true pose. Thirdly, $s_{\min}$ determines the minimum scaling value. If we are extremely confident in our map, then we can set this value to zero. This equals a complete rejection of corresponding pose estimates. We set it to $s_{\min} = 0.01$ for a strong weighting but not a complete rejection. Finally, we choose λ_2 such that the resulting function has a steep enough slope. With $\lambda_2 = 0.5$ the minimum scaling value is achieved for $|\delta| \geq -3.3\,\mathrm{m}$. Empirically this proves to be a reasonable



(a) Trajectory of the vehicle.



(b) Histogram of the distances to the center lines.

Figure 7.17.: Distance of the vehicle to the center line. This evaluation shows that the vehicle is usually driving close to the center line.

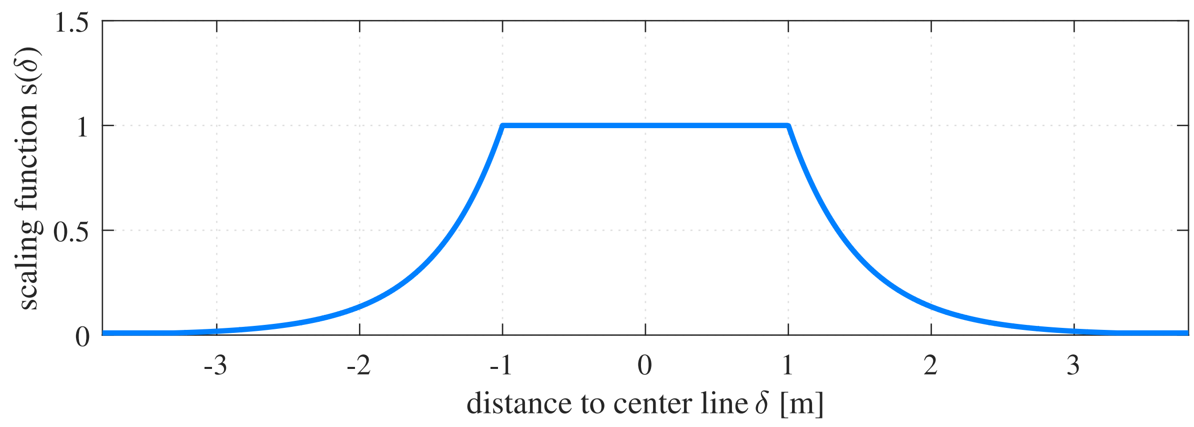


Figure 7.18.: Scaling function with experimentally determined parameters. We set $\lambda_1 = 1.0$, $\lambda_2 = 0.5$, $s_{\min} = 0.01$, and $s_{\max} = 1.0$.

value. Figure 7.18 shows a plot of the total resulting scaling function.

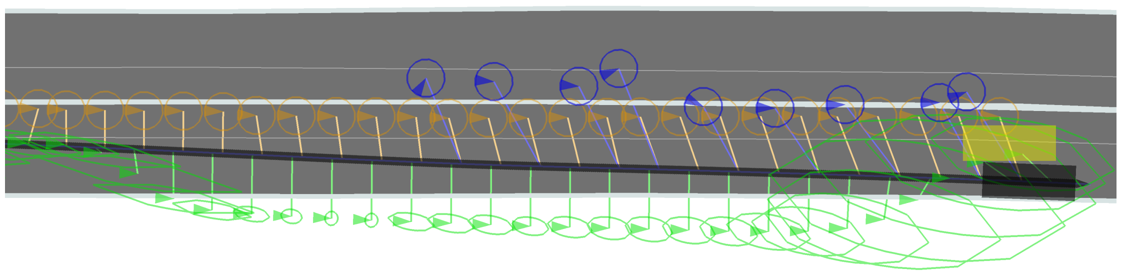
Having defined the scaling function, we now analyze the effects of its use. There are two major effects. The first one is that it selectively increases the uncertainty of pose estimates that seem a bit off. That is to say, most of the pose estimates are unscaled, but a couple every now and then are scaled because they are rather far away from the next center line. The second effect is that almost all pose estimates from a pose source are scaled. This happens when this pose source malfunctions, remains in its initialization phase, uses a map with an offset or a different coordinate system, or is simply generally of low accuracy. It is a nice property of this method that the user does not have to specify this manually. All corresponding pose estimates are automatically determined by comparing them to the center lines of a DLM.

We illustrate these two effects with an analysis of an interesting situation during a test run with the Audi A6 Avant. The short recording is about $40\,\mathrm{s}$ long and took place on a couple of hundred meters in an urban environment. We show the difference between not using map-based scaling, using it, and using a Pseudo-Huber cost function. This supports our claims that map-based scaling improves the pose fusion performance and that using a robust cost function can be counterproductive. In this analysis we use the core estimator with a temporal resolution of $\Delta t = 10\,\mathrm{ms}$, a maximum number of hidden nodes of $M = 1000$, and the scaling function as defined above. No other preprocessing modules or marginalization are used.

The input data consists of three different global and one local pose sources. To get a better idea of the driving situation at hand, we provide with Figure 7.19 two screenshots of the visualization of the PGF. The constraints of interest are those from the LBL. We note that they make a lateral jump at the beginning and at the end of the pictured section. The unscaled covariance matrices of its estimates show that the LBL considers this high-quality information. The covariance matrices during the jump are a lot bigger in comparison. We see in Figure 7.19a that the pose graph of the PGF is significantly bent towards LBL's constraints. As a result the estimated vehicle pose is off compared to the reference pose. In comparison, Figure 7.19b displays how the map-based scaling affects especially the covariance matrices of the LBL's pose estimates. As their distance to the nearest center line exceeds the threshold of λ_1, their covariance matrices get scaled considerably. This results in less weight on these constraints during the graph optimization. As a consequence, the pose graph of the PGF stays significantly closer to the pose estimates of the other pose sources. This also improves the final estimation of the vehicle pose.

Having inspected this snapshot, we now look at the pose fusion's position error over time with and without the map-based scaling module. Figure 7.20 shows the trajectory of the vehicle for the situation at hand. Qualitatively, the position error of the pose fusion decreases when using the map-based scaling module. The MAE drops from $1.47\,\mathrm{m}$ to $1.04\,\mathrm{m}$, because bad pose estimates are weighted less.

The map-based scaling shares similarities with robust cost functions. Both try to limit the influence of outliers in the optimization problem. Therefore, we evaluate our method against a Pseudo-Huber cost function. This is a standard and well-studied cost function that is applicable to many NLLSQ problems. Its disadvantage is, though, that it does not exploit all available information, such as maps. We rerun the PGF on the same data as above. This time we disable the map-based scaling and enable a Pseudo-Huber cost function on all constraints. We vary its parameter δ between $\delta = 0.5$ and $\delta = 2.0$. Table 7.8 summarizes the resulting position error of the pose fusion. Interestingly, the error increases for increasing values of δ. If the robust cost function affected mainly the observed nodes from the LBL, then we would expect the error to decrease. In this case, however, the justification mainly triggers on the observed nodes from the OEMStar receiver and the LiLoc. This is counterproductive and thus decreases the quality of the estimated pose. The underlying root cause here is that the Pseudo-Huber function simply evaluates the constraints' error values and ignores everything else. It is designed


(a) Without map-based scaling.

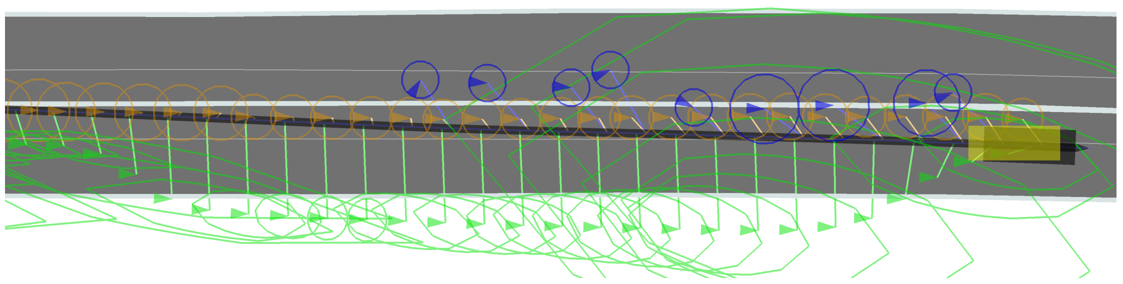
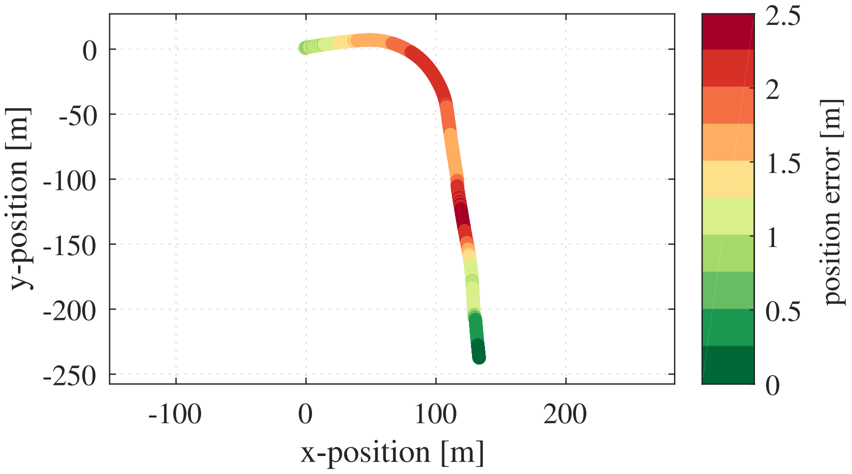
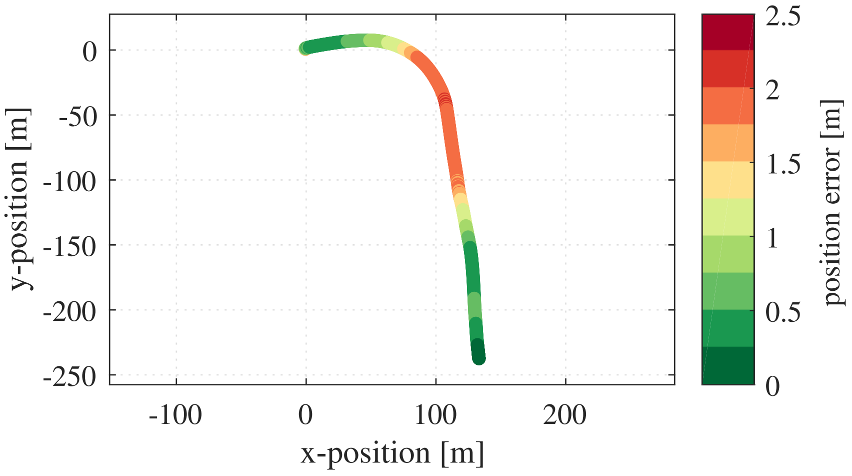

(b) With map-based scaling. The covariance matrices of the LBL's pose estimates are considerably larger.

Figure 7.19.: Situation analysis of the map-based scaling. The pose graph of the PGF (black line) is computed with and without map-based scaling of its input. The observed nodes of the LBL (green triangles) are subject to the most notable effect. The other two global pose sources are the OEMStar receiver (orange triangles) and LiLoc (blue triangles). The final estimated vehicle pose is shown as a black rectangle, while the reference pose is displayed as a yellow rectangle. The center lines are the two thin gray lines.



(a) Without map-based scaling.



(b) With map-based scaling.

Figure 7.20.: Position error of the PGF with and without map-based scaling. The error decreases when using the map-based scaling module. Figures best seen in color.

$\rho_{\mathrm{H}}(\cdot)$	median $[\mathrm{m}]$	MAE $[\mathrm{m}]$	RMS $[\mathrm{m}]$	maximum error $[\mathrm{m}]$	standard deviation $[\mathrm{m}]$
$\delta = 0.5$	1.82	1.53	1.75	2.62	0.86
$\delta = 1.0$	1.77	1.45	1.65	2.36	0.79
$\delta = 1.5$	1.73	1.42	1.60	2.32	0.76
$\delta = 2.0$	1.69	1.39	1.56	2.28	0.72

Table 7.8.: Statistics of the position error of the PGF when using different parametrizations of the Pseudo-Huber cost function $\rho_{\mathrm{H}}(\cdot)$.

to mitigate the influence of individual outliers. Here the outliers in form of the LBL constraints dominate the pose graph and thus get preferred. The key insight is that if additional semantic knowledge is available, it makes sense to exploit it. The map-based scaling provides a way to do that.

7.4.3. Correlated errors between pose sources

We account for correlated noise between input sources by applying a CI framework. The contribution consists in extending the state-of-the-art closed form solutions for the important case of covariance matrices with equal entries. We evaluate the approach with simulated data to support our claim that the treatment of unknown noise correlations between different input sources leads to conservative estimates. We also assess how often our extension to the CI framework is typically needed.

The experiment is designed to show that naive fusion produces overconfident estimates whereas CI produces conservative estimates. To this end, we generate a time series of random covariance matrices for a fixed cross-correlation $\rho = 0.6$ by sampling from a standard normal distribution, subsequently correlating the noise signals, and finally modifying them to have the desired mean and variance. With this data we perform naive fusion, optimal fusion, and both (trace and determinant minimizing) CI fusions, see Section 3.5 for more details. We check their estimated covariances over time and summarize the one-dimensional 3σ boundaries as boxplots. Figure 7.21 shows that the naive fusion is overly confident whereas CI produces uncertainty estimates equal to or greater than the optimal uncertainty. In this use case, CI with trace minimization produces similar results as CI with determinant minimization. The mean covariance of

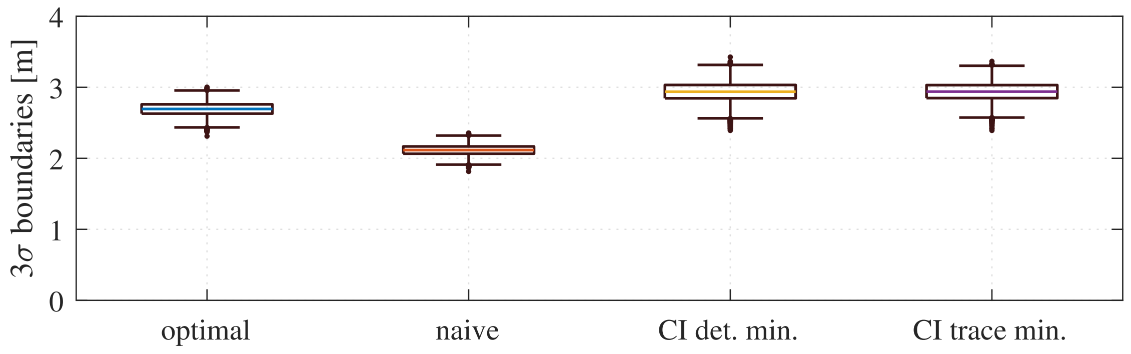


Figure 7.21.: Fusion of correlated covariance matrices with different methods for simulated data. Boxplots of the 3σ uncertainty boundaries of the estimate after fusion are shown. The naive fusion ignores the correlation and thus produces overconfident covariance estimates. The optimal fusion requires unavailable knowledge about the unknown correlation ρ. CI generates conservative estimates.

the optimal fusion is equal to $3\sigma_{\text{optimal}} = 2.70$. However, optimal fusion cannot be employed as the correlation is usually unknown. Both CI with trace and determinant minimization estimate on average a covariance of $3\sigma_{\text{CI}} = 2.94$, whereas the naive fusion estimates a mean covariance of $\sigma_{\text{naive}} = 2.10$. The proposed CI implementation therefore provides conservative estimates.

Furthermore, we evaluate how often the case of $\bar{d}_i = 1$ arises. In that situation, our extension to the CI framework comes into play. The frequency of this case depends strongly on the type of data and the threshold for the difference of two floating point numbers below which we consider both to be numerically identical. With errors as simulated in this experiment, the case of interest occurs in approximately $2\,\%$ to $5\,\%$ of all measurements.

In summary, we have shown that it is necessary to pay attention to correctly treat correlated errors. Ignoring them and fusing their data naively leads to overconfident estimates. The CI framework provides a suitable way for tackling the challenge of fusing estimates with correlated errors.

7.4.4. Autocorrelated errors

In this experiment we focus on our modeling of autocorrelated errors within a pose graph. We have seen in Section 6.4 that we can either model them with special constraints or, equivalently, by scaling the corresponding information matrices. We refer to the latter as *AR(1) scaling*. Its main effect is the improvement of the covariance matrix that the PGF computes for its resulting pose estimate. This experiment supports our claim that our modeling of autocorrelated errors leads to a better estimation of the uncertainty of the pose fusion result.

First, we need a metric to numerically grasp the concept of a "better uncertainty estimation". This metric should assess how well an estimated covariance matrix fits to the actual estimation error. For a single covariance matrix, this is difficult to say. By definition, the underlying Gaussian distribution corresponding to a covariance matrix extends in all directions infinitely. That is, all function values can possibly arise, albeit they are not equally likely. Therefore, we do not evaluate a single covariance matrix but instead calculate statistics over all of the covariance matrices computed by the PGF during a test run. The reasoning is that we have well-defined expectations about the behavior of the set of all covariance matrices. For one-dimensional Gaussian distributions, we expect for large sample sizes about $68.27\,\%$ of the data within the 1σ, $95.45\,\%$ within the 2σ, and $99.73\,\%$ within the 3σ boundary. Therefore, we check how well these expectations are met. For this we count how often the reference pose is contained within the 1σ, 2σ, or 3σ boundary of the estimated uncertainty. We call this metric the *uncertainty coverage*.

We conduct an experiment with simulated data and investigate on the uncertainty coverage of the PGF. Using simulated data allows us to ensure that the noise of each pose source is created by exactly the generating distribution that we want it to. We provide evaluations with real data in two experiments in Section 7.5. Here we use eight global and four local pose sources. All global pose sources create pose estimates at $5\,\mathrm{Hz}$ and all local pose sources at $100\,\mathrm{Hz}$. We add independent and identically distributed (i.i.d.) noise with $\sigma_x^{\mathrm{v}} = 0.0001\,\mathrm{m}$, $\sigma_y^{\mathrm{v}} = 0.0001\,\mathrm{m}$, and $\sigma_\theta^{\mathrm{v}} = 0.000\,01\,\mathrm{rad}$ to the odometry data. To the global pose estimates, we add noise with an AR(1) with $\phi_x^{\mathrm{w}} = \phi_y^{\mathrm{w}} = \phi_\theta^{\mathrm{w}} = 0.95$, $\sigma_x^{\mathrm{w}} = 3.00\,\mathrm{m}$, $\sigma_y^{\mathrm{w}} = 3.00\,\mathrm{m}$, and $\sigma_\theta^{\mathrm{w}} = 4.00°$. The PGF sets its number of hidden nodes to $M = 4000$ and its temporal resolution to $\Delta t = 10\,\mathrm{ms}$.

By construction, each pose source has a perfect uncertainty coverage because we know its generating distribution. We will see, though, that the PGF overestimates its confidence and provides covariance matrices with too small values. Why is that? This is caused by the mismatch between the core estimators noise assumptions and the pose sources' underlying noise distributions. The core estimator assumes its input data to have AWGN noise. The pose sources provide data with noise from an AR(1) model. Our AR(1) scaling provides a way to link those two together.

While the derivation of our modeling of autocorrelated errors within pose graphs seems admittedly complex, its application in the form of AR(1) scaling is straightforward. Given the number of observed nodes per source n within a pose graph, the scaling factor is computed using (6.114) to

$$\omega_i = \frac{n - (n-2)\phi}{n(1+\phi)}. \tag{7.1}$$

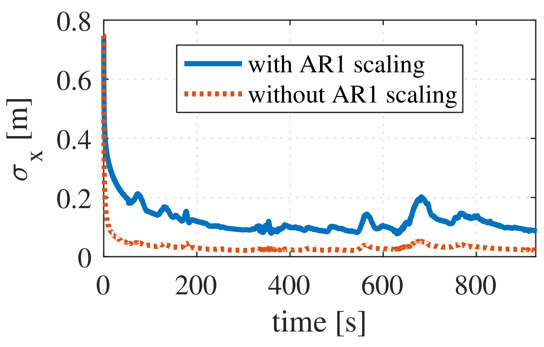
For our experiment the sliding window pose graph encompasses a time span of $M\Delta t = 10\,\mathrm{s}$ once it is built up. Observed nodes are added with $5\,\mathrm{Hz}$, leading to $n = \frac{10\,\mathrm{s}}{0.2\,\mathrm{s}} = 50$ observed nodes per source. Therefore, the scaling factor equals

$$\omega_i = \frac{50 - (50-2)0.95}{50(1+0.95)} \approx 0.045. \tag{7.2}$$

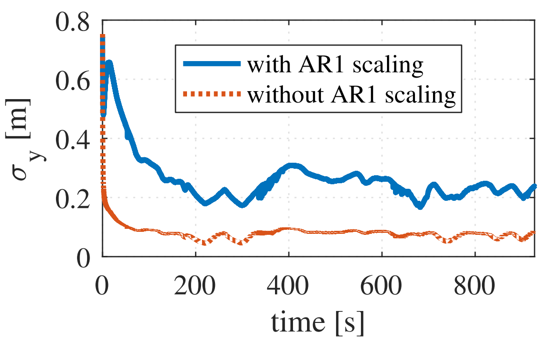
That is, the information matrices are scaled with a factor of ≈ 0.045, or in other words, the covariance matrices with a factor of $\frac{1}{\omega_i} \approx 22.16$.

We conduct the experiment with AR(1) scaling enabled and disabled. Additionally, we enable and disable marginalization. First, we examine the estimated uncertainty of the PGF. Figure 7.22 shows the evolution of the standard deviation in longitudinal direction, σ_x, and in lateral direction of the vehicle, σ_y. As desired, the AR(1) scaling results in an increased uncertainty of the output. However, does the estimated uncertainty better reflect the error?

To answer this question, we compute the uncertainty coverage of the PGF. The results for different configurations are provided in Table 7.9. Not making use of the AR(1) scaling leads to significantly too small uncertainty coverages. At most $49.10\,\%$ of the reference poses are contained within the 3σ boundary of the estimated covariance matrix. Enabling AR(1) scaling strongly increases the uncertainty coverage. It is about



(a)



(b)

Figure 7.22.: Standard deviation estimated by the PGF with and without AR(1) scaling, in (a) longitudinal and (b) lateral direction of the vehicle.

AR(1) scaling	marginalization	direction	1σ [%]	2σ [%]	3σ [%]
disabled	disabled	lateral	12.05	23.56	34.36
		longitudinal	12.24	22.55	31.90
	enabled	lateral	16.82	33.42	49.10
		longitudinal	11.84	24.57	34.91
enabled	disabled	lateral	54.10	88.47	96.44
		longitudinal	41.50	73.27	92.26
	enabled	lateral	58.47	90.14	99.08
		longitudinal	34.78	86.15	98.34

Table 7.9.: Uncertainty coverage of the PGF with and without AR(1) scaling. The numbers indicate how many percent of the reference poses lie within the corresponding uncertainty boundary.

three times larger than without. However, the coverage is not perfect, either. We attribute this to additional, unmodeled effects, such as interpolation, linearization, and timing effects.

Moreover, we note that the marginalization in form of the prior node works well together with the AR(1) scaling. Its direct influence on the estimated covariance matrix by adding another constraint is not the sole interaction. Instead, it influences the uncertainty coverage in another way by reducing the position error. If the error is smaller, then the reference pose is naturally closer to the estimated pose. We have seen this reduction of the position error when using the prior node already in the experiments in Section 7.3.2. In the experiments at hand, using the prior node decreases the MAE from $1.22\,\mathrm{m}$ to $0.35\,\mathrm{m}$ when using AR(1) scaling. In the experiments with AR(1) scaling disabled, using the prior node has a comparable effect and decreases the MAE from $1.17\,\mathrm{m}$ to $0.42\,\mathrm{m}$.

All in all, this experiment has shown that the AR(1) scaling increases the PGF's estimated covariance matrix, that it works nicely together with the prior node, and that it reduces the overconfidence of the estimation.

7.5. Pose fusion layer

In this section we provide experiments to evaluate the entire pose fusion layer, which comprises both the preprocessing sublayer and the core estimator. We carry out experiments on three datasets that we obtained on real prototype vehicles. The experiments are conducted in different scenarios, with different vehicles, and with different pose sources to highlight the versatility of the pose fusion.

The first experiment in Section 7.5.1 is conducted in a rural and urban environment with vehicle speeds up to $100\,\frac{\mathrm{km}}{\mathrm{h}}$. Four different global pose sources and two local pose sources are fused. This demonstrates that we can successfully fuse many heterogeneous sources.

The second experiment in Section 7.5.2 is performed in an urban setting with a maximum speed limit of $30\,\frac{\mathrm{km}}{\mathrm{h}}$. Two global and one local pose source are fused. We show the fusion performance with a low number of input sources.

Lastly, we present in Section 7.5.3 an experiment that we performed during a drive through a parking garage. This leads to an extended period of dead reckoning.

7.5.1. Experiment on an Audi A6

This experiment is designed to evaluate our claims that the combination of preprocessing sublayer and core estimator can effectively fuse data gathered on a real prototype vehicle. For this we show the impact of the preprocessing sublayer on the input data and the pose fusion. We support our claim that the bias estimation leads to a reduction of the pose fusion's error. Also, our modeling of autocorrelated errors reduces the overconfidence of the estimator.

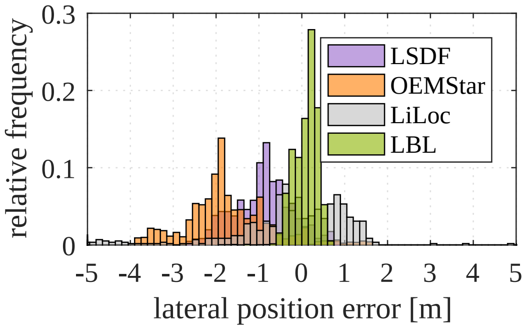
The vehicle that we use for this experiment is the Audi A6 presented in Section 7.1. We demonstrate that the pose fusion can handle data from four global and two local input sources. The global pose sources are LSDF (see Section 7.2.4), an OEMStar receiver (see Section 7.1.1), LiLoc (see Section 7.2.3), and LBL (see Section 7.2.2). The two local pose sources are EgoMaster (see Section 7.2.1) and LSDF relative (Section 7.2.4). We use the Applanix system as reference (see Section 7.1.2).

The data for the experiment is gathered during a drive of about 21 minutes in rural and urban areas in Germany. We begin by inspecting the data of the pose sources. This gives us an overview over their performances. After that, we test the quality of the pose fusion without preprocessing sublayer. This serves as a comparative value later on. Subsequently, we evaluate the influence of the preprocessing sublayer on the data of the input sources. Finally, we plug the preprocessing sublayer and the core estimator together and analyze their performance.

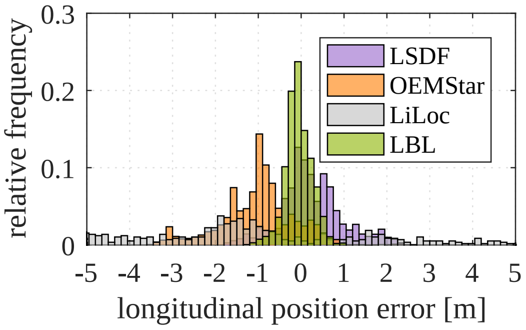
Analysis of the input pose estimates

As a first step we analyze the data of all input sources. We compare their data to our reference system and provide statistics that summarize the evaluation. We start with the global pose sources and address the local pose sources afterwards.

At first, let us take a closer look at the position error of the global pose sources. To get a better insight into the their performance, we distinguish between the lateral and longitudinal position error rotated into the reference pose's frame. The histogram of the errors in displayed in Figure 7.23. Without further quantitative analysis, we already state that LBL pose source is the most accurate one. Both the lateral and longitudinal error are centered closely around zero. The position errors of the other pose sources are clearly larger. If a histogram shows that most of the data exhibits a positive lateral (or



(a)



(b)

Figure 7.23.: Histogram of the position error of the four global pose sources.

source	direction	mean [m]	median [m]	MAE [m]	RMS [m]	standard deviation [m]
LSDF	lateral	-0.91	-0.87	0.98	1.14	0.69
	longitudinal	0.25	0.16	0.57	0.76	0.72
OEMStar	lateral	-1.73	-1.84	1.80	1.99	1.00
	longitudinal	-1.02	-0.96	1.11	1.33	0.85
LiLoc	lateral	-0.60	-0.15	1.34	2.24	2.16
	longitudinal	-2.74	-1.95	3.85	4.99	4.17
LBL	lateral	0.10	0.15	0.24	0.28	0.26
	longitudinal	-0.06	-0.08	0.24	0.31	0.30

Table 7.10.: Statistics of the position error of the input sources.

longitudinal) position error, that means that the estimated position is generally too far to the left (or to the front) of the true position.

To judge the quality of the input sources on a more numerical basis, we compute some statistics over their position errors. Table 7.10 summarizes them. As with the histograms, we find that LBL performs the best. It generally has the lowest error statistics.

The core estimator is built for input data with AWGN characteristics. In other words the input data should be unbiased, subject to Gaussian noise, and independently generated. These conditions are more or less strongly violated. From Table 7.10 we see that the mean position errors of LSDF, the OEMStar receiver, and LiLoc are significantly different from zero. Therefore, they are subject to systematic biases. In Figure 7.23 we see that the position error of none of the pose sources is purely dominated by a Gaussian distribution. The condition that demands independently generated samples is also violated. It implies that the time series of the input data for each source is not correlated over time. We analyze the partial autocorrelation function (PACF) of each time series to show that all input pose sources are autocorrelated. For a given time series x_t the partial autocorrelation between samples x_t and x_{t-l} is the conditional correlation between x_t and x_{t-l} conditioned on the set of observations that lie between the time steps t and $t-l$. In this sense, the PACF controls for the values of the time series at all shorter lags. This is in contrast to the autocorrelation function, which does not control for other lags. Figure 7.24 displays the PACF for all global input sources' position errors. Clearly, all time series exhibit at least a strong autocorrelation of order 1. That means the position error of time step t is strongly correlated with the position error at $t-1$. This is a side effect of temporal filtering, for example. We conclude that the position errors cannot be explained by assuming AWGN. As a remedy for the violated noise assumptions, we will later use the appropriate modules of the preprocessing sublayer.

We turn our attention to the local pose sources EgoMaster and LSDF relative. Again, we determine their estimation errors. As these pose sources generate estimates about the current movement of the vehicle, we cannot directly apply the same statistical tool set as for the global pose sources. Instead, we compute their errors as the Euclidean difference between their estimated movement and the actual movement in the reference system's frame. Figure 7.25 shows the EDF of their errors. We have downsampled the EgoMaster data from $100\,\mathrm{Hz}$ to $50\,\mathrm{Hz}$ to make it comparable on a per sample basis to LSDF relative's. As the output frequencies of these sources are rather high, the absolute magnitude of their estimated movements as well as their errors for a single time step are small. The errors naturally add up over time, though. Roughly $50\,\%$ of LSDF relative's output is significantly more noisy than EgoMaster's.

Another common measure to describe the accuracy of a local pose source is to compute the ratio of its movement error with respect to the driven distance. This gives an intuition about the accuracy without going into too much details. However, the measure is problematic whenever the vehicle moves very little. It is even undefined for standstill phases. Therefore, we refrain from presenting a detailed analysis. Instead we simply state that both local pose sources generally exhibit a movement error of below $0.5\,\%$ of

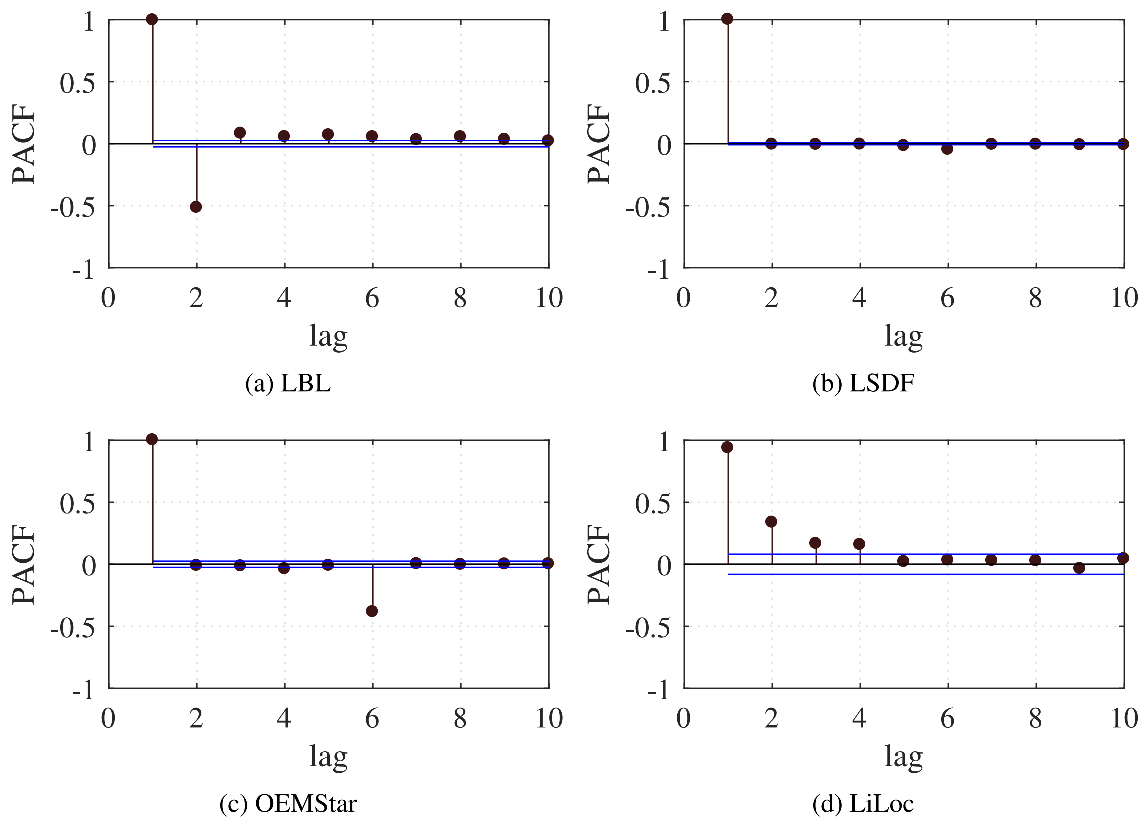


Figure 7.24.: PACF of the position error of the global input sources. The blue lines indicate the 5% significance limits for the partial autocorrelations.

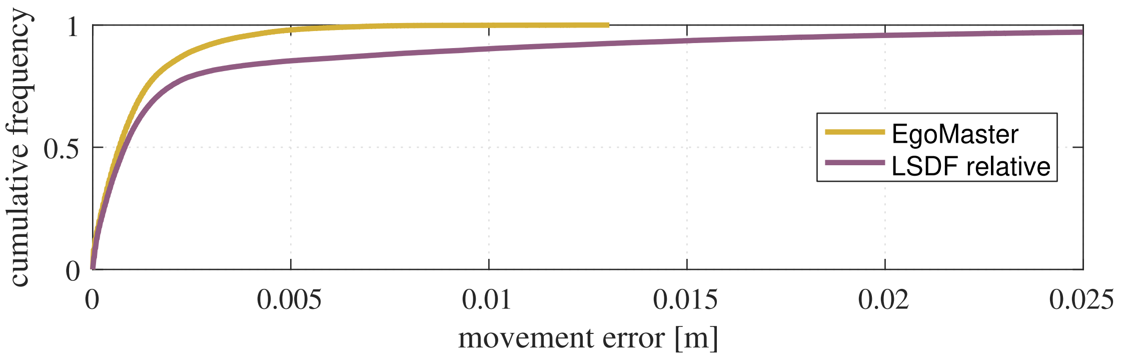


Figure 7.25.: EDF of the movement error in the reference frames of the local pose sources.

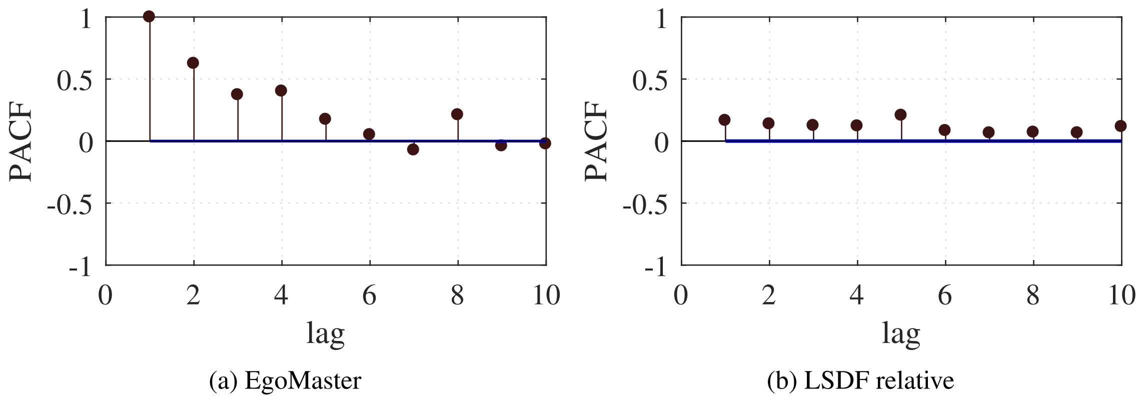


Figure 7.26.: PACF of the position error of the local input sources. The blue lines indicate the 5% significance limits for the partial autocorrelations. They are very close to zero due to the large number of samples.

the driven distance.

As we did for the global pose sources, we also examine the PACF for the local pose sources. Figure 7.26 shows the resulting plots. We find strong autocorrelations which we attribute to a temporal filtering in their algorithms. However, this is an assumption because source code or a detailed description of them is not available.

Next, we inspect the nominal frequencies and the availability of the input sources. The nominal frequency is the frequency at which a pose source is supposed to output samples. We define the availability as the ratio of the number of samples that we received over the number of samples that we should have received. For the latter we multiply the nominal output frequencies of the pose sources with the duration of the dataset.

Table 7.11 lists the nominal frequencies and the availability of the input sources. We note that their nominal frequencies are spread over a broad range between $2\,\mathrm{Hz}$ to $100\,\mathrm{Hz}$. Additionally, the availability shows that LiLoc is only sparsely available whereas data from the OEMStar receiver and EgoMaster is almost always available. All other pose sources range in between. This means that they occasionally drop out. This happens because they sometimes do not have map data available, their underlying algorithms cannot compute a pose estimate given their input data, or their calculations take too long. As a consequence, the PGF has to handle different input frequencies and data dropouts.

input source	nominal frequency $[\mathrm{Hz}]$	availability $[\%]$
LSDF	50	82.69
OEMStar	5	100.00
LiLoc	2	24.07
LBL	5	94.56
LSDF relative	50	83.35
EgoMaster	100	99.99

Table 7.11.: Availability and nominal frequencies of the input sources.

Pose fusion performance without the preprocessing sublayer

We start our analysis of the pose fusion by directly plugging the pose sources into the core estimator. We deliberately skip as much of the preprocessing sublayer as possible such that we can later clearly identify its influence. It is not possible to skip it entirely as we need to set at least covariance matrices for pose sources which do not provide any for their estimates. These third-party black box modules without proper uncertainty estimation include the OEMStar receiver and LiLoc. For their data we make use of their sample covariance matrices as determined empirically during previous test runs. This is of course not optimal, but further information is unavailable.

Figure 7.27 illustrates a small portion of the pose graph that is being built up during the experiments. Comparing it to Figure 7.6 we clearly see the differences between the simulated and the real input data. In contrast to the simulated data, the effect of autocorrelation is strongly visible for all global input sources. Moreover, this small portion of the graph already gives us a visual hint about existing biases in the data. Additionally, we get an impression of the different frequencies and latencies which the pose sources exhibit. All of these effects have already been highlighted in a quantitative way in our preceding analysis of the input data.

We set up the PGF with common values for processing this kind of data. These include setting the temporal resolution to $\Delta t = 10\,\mathrm{ms}$, the maximum number of hidden nodes to $M_{\mathrm{max}} = 1000$, and the PID parameters to $K_{\mathrm{p}} = 2$, $K_{\mathrm{i}} = 200$, and $K_{\mathrm{d}} = 2$. The output frequency is set to $f = 20\,\mathrm{Hz}$ and marginalization is turned off.

The distribution of the position error of the PGF is shown as two histograms in Figure 7.28. We note that the error is rather widespread and not centered around zero. Ta-

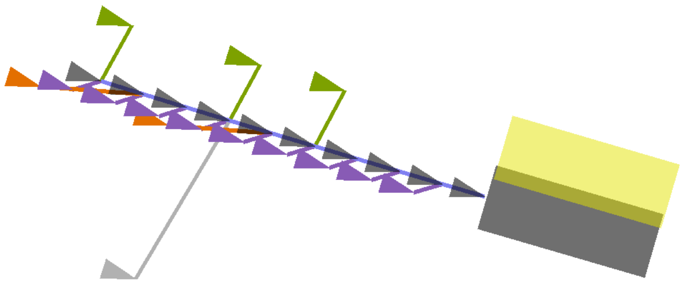


Figure 7.27.: Illustration of the core estimator's pose graph. The gray nodes visualize its hidden nodes, and the colored nodes are observed nodes stemming from different global pose sources (green: LBL; orange: OEMStar; purple: LSDF; light gray: LiLoc). The yellow rectangle shows the true pose of the vehicle, while the gray rectangle shows the estimated pose.

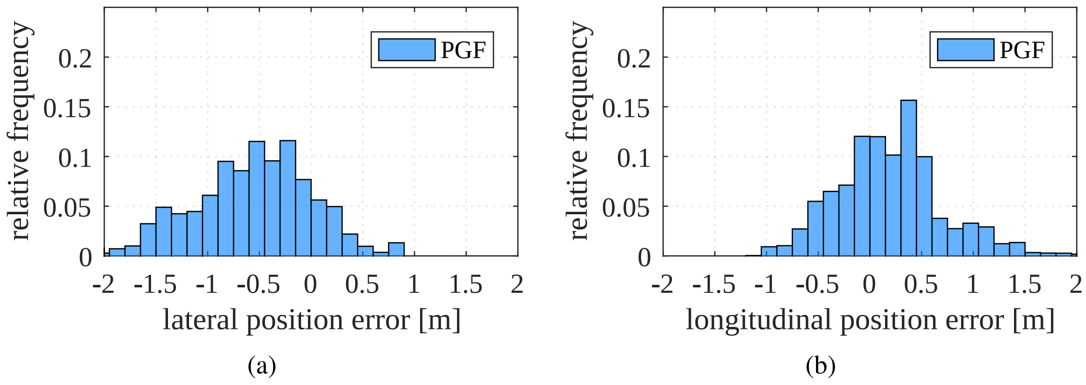


Figure 7.28.: Histogram of the position error of the PGF.

error component	mean $[\mathrm{m}]$	median $[\mathrm{m}]$	MAE $[\mathrm{m}]$	RMS $[\mathrm{m}]$	standard deviation $[\mathrm{m}]$
lateral	-0.58	-0.50	0.66	0.82	0.58
longitudinal	0.20	0.16	0.40	0.55	0.51
Euclidean	0.86	0.78	0.86	0.99	0.49

Table 7.12.: Statistics of the position error of the PGF.

uncertainty coverage $[\%]$	1σ	2σ	3σ
lateral	4.30	7.87	11.93
longitudinal	9.17	16.40	22.65

Table 7.13.: Uncertainty coverage of the PGF. The core estimator underestimates its actual uncertainty without the preprocessing sublayer.

ble 7.12 presents the error statistics. The mean errors with $-0.58\,\mathrm{m}$ and $0.20\,\mathrm{m}$ clearly show that the systematic errors of the input sources strike through on the fusion result.

Next, we check the quality of the uncertainty coverage of the PGF. We expect this measure to be negatively influenced by both the autocorrelated errors of the input sources and the systematic bias of the fusion result. These two effects theoretically lead to an overconfident estimator. The results presented in Table 7.13 confirm this. Only $11.93\,\%$ or $22.65\,\%$ of the estimated poses contain the reference pose within the lateral or longitudinal covariance boundary of 3σ. This means that the actual uncertainty is massively underestimated. The estimator is therefore overly confident without the preprocessing sublayer.

Effect of the preprocessing sublayer

In the previous section we have seen the pose fusion result without using the preprocessing sublayer. Three main problems arose: the input sources provide data with systematic biases, autocorrelated errors, and unsuitable or missing covariance matrices. Our preprocessing sublayer offers three modules to counteract these problems. Those are the bias estimation, the autocorrelated noise modeling, and the map-based outlier handling. One by one we analyze their application.

The bias estimation module aims at minimizing time-varying biases. It is based on

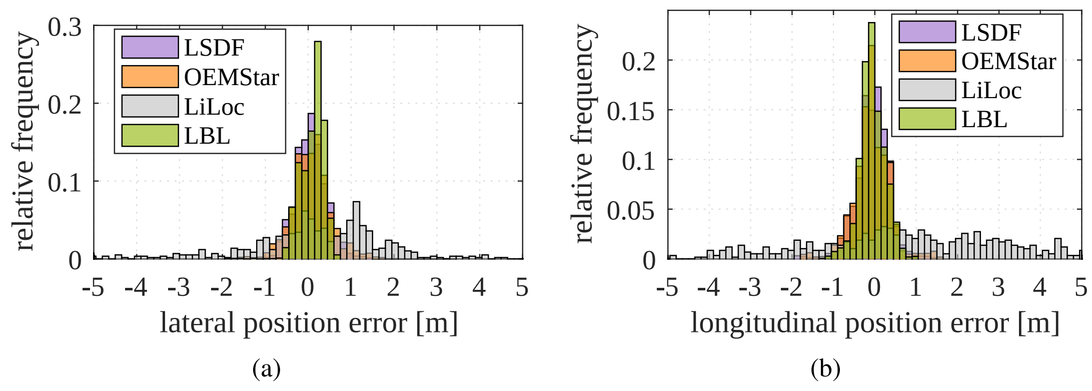


Figure 7.29.: Histogram of the position error of the four global pose sources with bias estimation. The error distribution for all pose sources is centered more around zero than without bias estimation.

comparing biased pose estimates to unbiased ones within a sliding window. For this we need to identify an unbiased pose source. We have seen in Figure 7.23 and Table 7.10 that the pose source with the overall least mean error is LBL. Therefore, we employ it as pose source against which we compare the other pose source's data. The comparisons are done separately from each other. We use a sliding window length of $15\,\mathrm{s}$.

After applying the bias estimation, we again compute the position error of all global pose sources. Figure 7.29 shows the histograms of the position errors. The position errors of LSDF and the OEMStar receiver are much more closely centered around zero compared to Figure 7.23. The distribution of the position errors of LiLoc, however, is rather flat and benefits the least from the bias estimation.

Table 7.14 underlines these observations. We note that the error statistics of the LBL stay of course untouched. The mean position errors of all other pose sources have significantly dropped. This means that we successfully estimated their biases. Additionally, the MAE of LSDF and the OEMStar receiver are considerably reduced. They are now within the same order of magnitude as the LBL's. Interestingly, also their standard deviations decreased around $50\,\%$. Only the standard deviation of the lateral error of LiLoc increased slightly from $2.16\,\mathrm{m}$ to $2.40\,\mathrm{m}$. These enormous gains in accuracy highlight the benefit of our bias estimation technique on real world data.

Next, we apply the autocorrelated noise modeling. We have seen in our previous analysis that all input sources show signs of filtered data or at least autocorrelated errors.

source	direction	mean [m]	median [m]	MAE [m]	RMS [m]	standard deviation [m]
LSDF	lateral	0.06	0.05	0.28	0.36	0.36
	longitudinal	-0.06	-0.03	0.30	0.40	0.40
OEMStar	lateral	0.07	0.06	0.35	0.46	0.46
	longitudinal	-0.08	-0.09	0.32	0.45	0.44
LiLoc	lateral	-0.30	0.14	1.61	2.42	2.40
	longitudinal	0.97	0.84	2.35	3.06	2.90
LBL	lateral	0.10	0.15	0.24	0.28	0.26
	longitudinal	-0.06	-0.08	0.24	0.31	0.30

Table 7.14.: Statistics of the position error of the pose sources.

Modeling them as AR(1) process might not be fully adequate, but is substantially more appropriate compared to ignoring the autocorrelations completely. If the errors could truly be modeled by an AR(1) process, then all coefficients greater than the first one should be close to zero.

Again, we split up the lateral and longitudinal error components of the position error for each pose source. We model each component separately as an AR(1) process. From the PACF plots we read off the coefficients at the first lag. For LBL, the OEMStar receiver, and LSDF, they amount to $\phi = 0.99$ or even slightly above, but we limit the value at 0.99. For LiLoc the lateral coefficient amounts to 0.93 and the longitudinal one to 0.90. Using these values we scale the covariance matrices of the input pose estimates accordingly. Consider the pose estimates from the LBL as an example. After the initial built up of the sliding window chain pose graph of the PGF there are usually $n = M \cdot \mathit{\Delta} t \cdot f_{\text{LBL}} = 1000 \cdot 0.01\,\text{s} \cdot 5\,\text{Hz} = 50$ observed nodes from the LBL within the graph. The according scaling factor is equal to

$$\omega_i = \frac{n - (n-2)\phi}{n(1+\phi)} = \frac{50 - (50-2)0.99}{50(1+0.99)} \approx 0.025. \tag{7.3}$$

The covariance matrices are therefore scaled by a factor of roughly $\frac{1}{0.025} = 40$. We will see the effects of that in the next evaluation of the pose fusion.

The last preprocessing module of interest is the map-based outlier handling for global

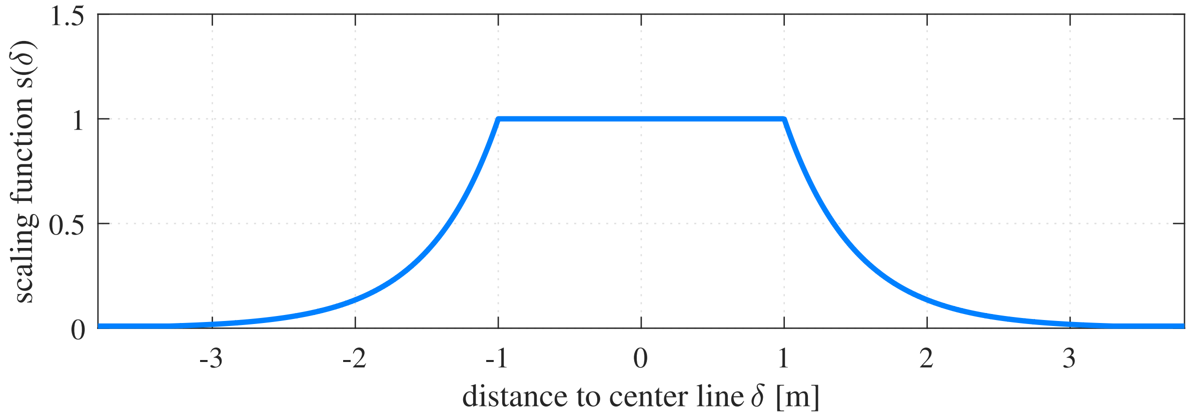


Figure 7.30.: Scaling function for different distances to the center line used during this experiment.

pose sources. It introduces prior knowledge in the form of a map to the covariance estimation. This knowledge is used to compare pose estimates to a DLM map. If they are far off any center line, the uncertainty of that pose estimate is scaled up or, more precisely, its information matrix is scaled down. We parametrize the scaling function $s(\delta)$ rather strictly with $s_{\mathrm{min}} = 0.01$, $s_{\mathrm{max}} = 1.0$, $\lambda_1 = 1.0$, and $\lambda_2 = 0.5$. Figure 7.30 shows a plot of the resulting function.

We apply this to all four global pose sources. Interestingly, it has different effects on them. It serves to downvote single outlier pose estimates for LBL, LSDF, and the OEMStar receiver. Most of their pose estimates are within a distance of $-1\,\mathrm{m}$ to $1\,\mathrm{m}$ to the center line. Therefore, their information matrices are not scaled because for $\delta \in [-1, 1]$ we have $s(\delta) = 1$. In contrast, most of the pose estimates of LiLoc are far away from the center line: its lateral MAE is $1.61\,\mathrm{m}$ after all. To investigate on this intuition, we check how much of the pose estimates are within a distance of $-1\,\mathrm{m}$ to $1\,\mathrm{m}$ to the center line. Table 7.15 breaks the results down per pose source. As expected, pose estimates from LiLoc are with $45.42\,\%$ the least frequently close to the center line. More than half of its pose estimates' information matrices are scaled down. Additionally, we note that the pose estimates from the reference system Applanix are in $94.70\,\%$ of the cases close to the center line. The remaining fraction is due to lane changes, corner cutting maneuvers, or inaccuracies within the map.

All in all, the preprocessing modules have a strong influence on the pose estimates. Figure 7.31 provides a visual impression of the effects. Comparing it to Figure 7.27 we

source	fraction of pose estimates within a distance of $1\,\mathrm{m}$ to the center line $[\%]$
Applanix	94.70
LSDF	86.93
OEMStar	81.87
LiLoc	45.42
LBL	91.83

Table 7.15.: Fraction of pose estimates close to the center line per pose source.

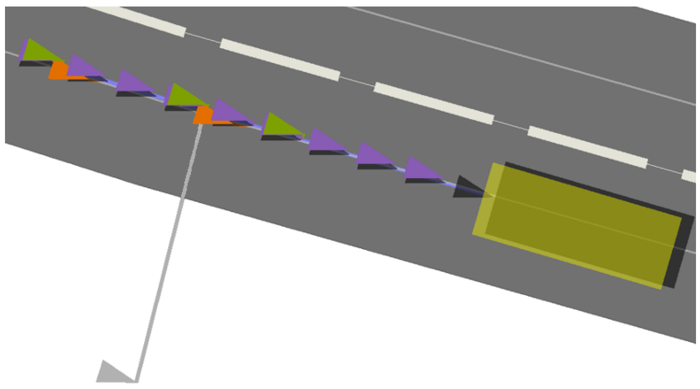
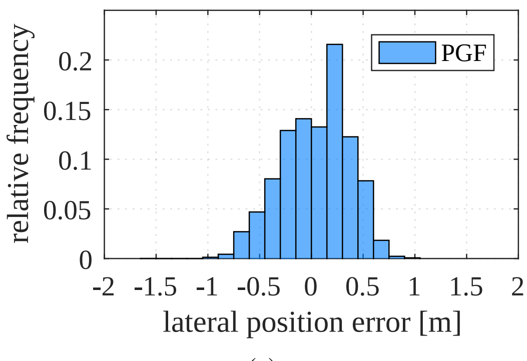


Figure 7.31.: Illustration of the core estimator's pose graph when using the preprocessing sublayer. The observed nodes are much less biased.

observe that the outlier handling module leads to the usage of a map within the system. Information matrices of poses far off the center line, like the light gray one, are scaled up (not shown). The pose estimates from the other sources agree much closer on the recent trajectory of the vehicle as their biases have been largely removed. This data forms the input for our next evaluations.

Pose fusion performance with preprocessed input data

We apply the preprocessing sublayer to the data from the pose sources and analyze the pose fusion performance of the PGF again. Figure 7.32 shows the resulting histograms of the position error. Qualitatively, it is centered more closely around zero than before. The numerical evaluations provided in Table 7.16 confirm this. The previous lateral and longitudinal mean errors decrease from $-0.58\,\mathrm{m}$ and $0.20\,\mathrm{m}$ to $0.04\,\mathrm{m}$ and $0.01\,\mathrm{m}$, respectively. This shows the effectiveness of the bias estimation. The MAE also drops



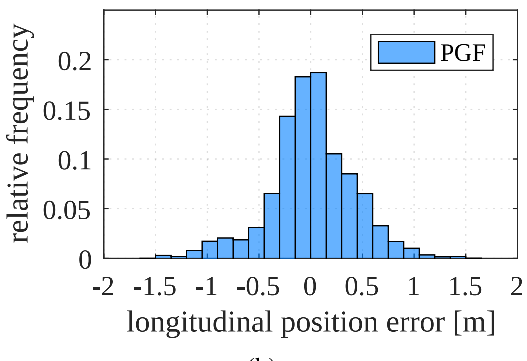


Figure 7.32.: Histogram of the position error of the PGF with preprocessed input data. The error is much closer distributed around zero for both the lateral and longitudinal error component compared to the fusion output without preprocessed input data (see Figure 7.28).

error component	mean $[\mathrm{m}]$	median $[\mathrm{m}]$	MAE $[\mathrm{m}]$	RMS $[\mathrm{m}]$	standard deviation $[\mathrm{m}]$
lateral	0.04	0.08	0.27	0.33	0.33
longitudinal	0.01	0.01	0.30	0.41	0.41
Euclidean	0.45	0.39	0.45	0.52	0.26

Table 7.16.: Statistics of the position error of the PGF with preprocessed input data. The error improves substantially.

significantly from $0.66\,\mathrm{m}$ and $0.40\,\mathrm{m}$ to $0.27\,\mathrm{m}$ and $0.30\,\mathrm{m}$ in lateral and longitudinal direction. Even the standard deviation decreases from $0.58\,\mathrm{m}$ and $0.51\,\mathrm{m}$ to $0.33\,\mathrm{m}$ and $0.41\,\mathrm{m}$, showing that the error is not only shifted but indeed reduced.

Finally, we examine the uncertainty coverage of the output pose estimates of the PGF. Table 7.17 summarizes the results. Comparing the numbers to the previous results without using the preprocessing sublayer given in Table 7.13, we note a strong improvement due to our modeling of autocorrelated errors. The true vehicle pose is much more often than before within the estimated uncertainty ellipses. However, the values are still below what we would expect from a true Gaussian estimator. We mostly attribute this to the assumption that we can perfectly model all pose sources as having errors from an AR(1) process.

All in all, we have shown in this experiment that the PGF can effectively fuse data

direction	1σ [%]	2σ [%]	3σ [%]
lateral	60.09	86.09	93.54
longitudinal	39.61	68.13	81.89

Table 7.17.: Uncertainty coverage of the PGF with preprocessed input data. The estimated uncertainty reflects the actual error significantly better than without the preprocessing sublayer.

from many different input sources on a real prototype vehicle. Additionally, we have demonstrated that the preprocessing sublayer provides powerful means for improving the input data in a way that makes it compatible with the core estimator. After its application, we have shown that the resulting accuracy is comparable to the accuracy of the best performing input source. Also, the preprocessing sublayer considerably decreases the overconfidence of the estimator.

7.5.2. Experiment on an e-Golf

This experiment is performed on another prototype vehicle to prove our claim that the PGF successfully runs on different vehicles. For this we use the e-Golf presented in Section 7.1. Also, changing the vehicle means limiting the number of input sources. Thus, we examine the pose fusion performance when the PGF has only two global and a local pose source as input. Moreover, using a different vehicle also means changing the input sources. Compared to the experiment in Section 7.5.1, there is no OEMStar but instead a u-blox receiver (see Section 7.1.1). The reference system also changes from the Applanix to the Oxford system (see Section 7.1.2).

In this experiment one focus is on the timing aspects. We prove our claims that the PGF handles different and time-varying input frequencies, deals with different latencies of the input poses, provides recent pose estimates at a near-constant output frequency, and is highly available. Additionally, we show interesting, non-intuitive behavior of the PGF that does not always lead to optimal results. We believe that presenting these kind of results leads to valuable insights over the pose fusion's characteristics. For this we provide in-depth analysis of selected route sections.

The presentation of this experiment is constructed similarly to Section 7.5.1. At first, we analyze the input data, then take a closer look at the preprocessing sublayer, and

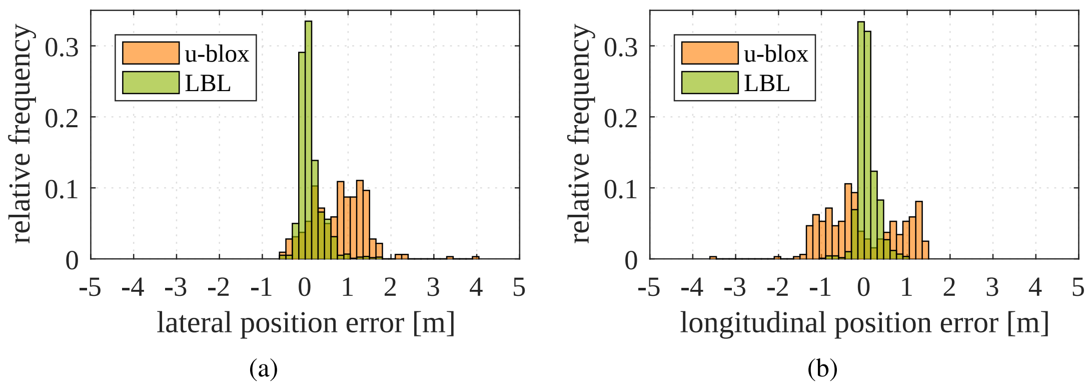


Figure 7.33.: Histogram of the position error of the two global pose sources.

source	direction	mean $[\mathrm{m}]$	median $[\mathrm{m}]$	MAE $[\mathrm{m}]$	RMS $[\mathrm{m}]$	standard deviation $[\mathrm{m}]$
u-blox	lateral	0.83	0.82	0.86	1.02	0.60
	longitudinal	-0.05	-0.24	0.72	0.84	0.84
LBL	lateral	0.12	0.05	0.18	0.29	0.26
	longitudinal	0.06	0.03	0.15	0.22	0.22

Table 7.18.: Statistics of the position error of the two input sources.

finally plug them into the core estimator.

Analysis of the input pose estimates

To start off our analysis, we analyze the data of the global pose sources. As a second step, we extend this to the local pose source. Figure 7.33 shows the histogram of the position error of the two global input sources. Both the lateral and longitudinal position error of LBL's pose estimates are centered around zero. The distribution of the lateral position error of the u-blox receiver's data is displaced compared to the LBL's. Table 7.18 provides numerical results. They show that the lateral mean error of the u-blox receiver's data is indeed significantly different to zero with a value of $0.83\,\mathrm{m}$. The distribution of the position error of the LBL is fairly well free of systematic biases. Also, the standard deviation of its error is considerable smaller than the u-blox receiver's data.

input source	nominal frequency [Hz]	availability [%]
u-blox	1	100.00
LBL	5	73.33
EgoMaster	100	100.00

Table 7.19.: Availability and nominal frequencies of the input sources.

We analyze the autocorrelation of the pose source's errors. Not surprisingly we again find that the errors of the pose estimates of the LBL are strongly autocorrelated. Also unsurprisingly we find the same for the u-blox receiver's data. It comes with an integrated IMU and the manufacturer promotes it with filtering and dead reckoning functionalities. The filtering leads of course to autocorrelated errors. Similar to the last experiment we conclude that the noises of both global pose sources cannot be described as AWGN.

Next, we briefly look at the timing of all pose sources. Table 7.19 summarizes the nominal frequencies and availability of the input sources. In contrast to the last experiment we find that the LBL is only available in $73.33\,\%$ of the time. We will show later in Figure 7.42 that its actual frequency is differing in roughly $20\,\%$ of the time from its nominal frequency. Also, its input data commonly has a latency between $100\,\mathrm{ms}$ to $300\,\mathrm{ms}$.

We now turn to the local pose source, the EgoMaster. The EDF of its movement error is displayed in Figure 7.34. Interestingly, the curve has a different form than in the previous experiment, see Figure 7.25. Whereas there were already approximately $78\,\%$ of samples with a movement error below $0.0015\,\mathrm{m}$, there are now only approximately $14\,\%$ of samples at the same value. The maximum movement error also increases from roughly $0.013\,\mathrm{m}$ by $14\,\%$ to $0.0148\,\mathrm{m}$. Indeed, the entire curve seems translated to the right. We attribute this to a systematic error in the movement estimation of the EgoMaster on this prototype vehicle. It is not constant over different test drives. Also, we do not have another local pose source with which we could use a similar technique to our bias estimation for global pose sources. Therefore, we have unfortunately little means to eliminate this systematic error. We will see that the pose fusion suffers from this.

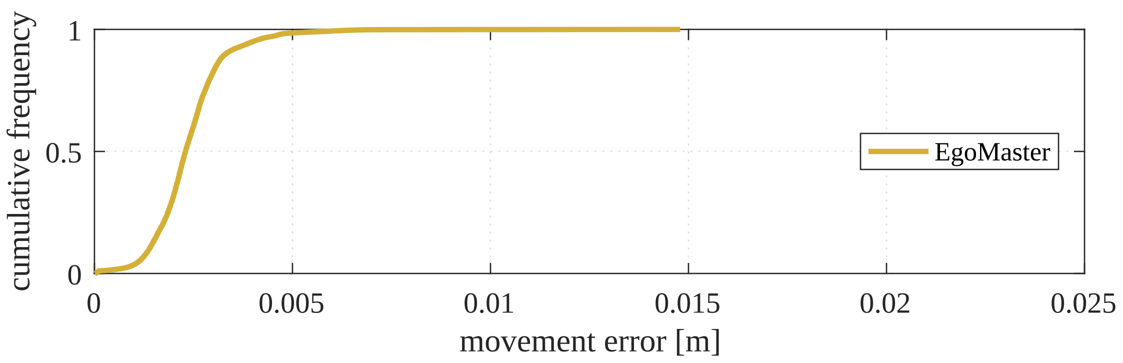


Figure 7.34.: EDF of the movement error of the EgoMaster.

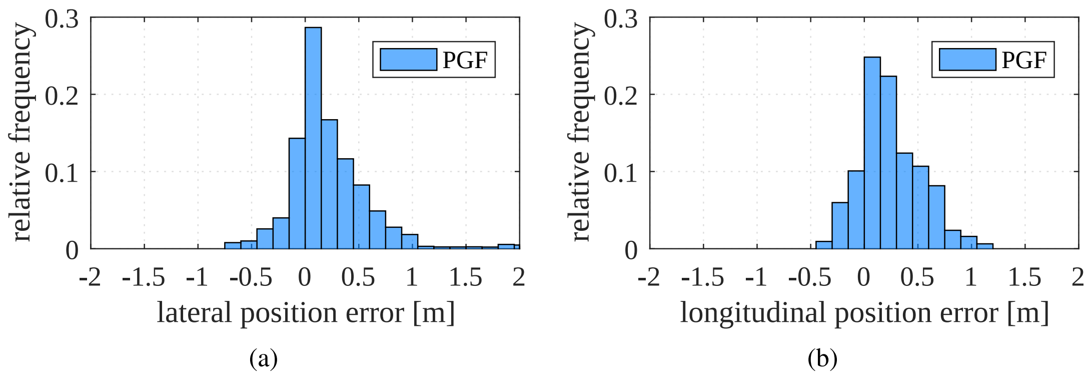


Figure 7.35.: Histogram of the position error of the PGF without the preprocessing sublayer.

Pose fusion performance without the preprocessing sublayer

We plug in the data streams from the pose sources into the core estimator without making use of the preprocessing sublayer. The core estimator is parametrized in the same way as in the previous experiment. The histogram of the position error is displayed in Figure 7.35. The corresponding statistics are given in Table 7.20. They both show an offset of the lateral and longitudinal mean errors. We attribute the longitudinal bias for the most part to the systematic EgoMaster error.

As a next step we analyze the uncertainty coverage. Table 7.21 summarizes the results. We observe that the reference pose is only in $50.36\,\%$ and $27.79\,\%$ within the 3σ-ellipse of the PGF. In other words, the core estimator is overconfident in its result. However, we notice a strong increase in the uncertainty coverage compared to the pre-

error component	mean $[\mathrm{m}]$	median $[\mathrm{m}]$	MAE $[\mathrm{m}]$	RMS $[\mathrm{m}]$	standard deviation $[\mathrm{m}]$
lateral	0.22	0.14	0.30	0.44	0.38
longitudinal	0.24	0.20	0.29	0.37	0.28
Euclidean	0.47	0.41	0.47	0.58	0.33

Table 7.20.: Statistics of the position error of the PGF without the preprocessing sublayer.

σ coverage $[\%]$	1σ	2σ	3σ
lateral	21.30	37.76	50.36
longitudinal	7.40	17.66	27.79

Table 7.21.: Uncertainty coverage of the PGF without the preprocessing sublayer.

vious experiment. This is due to using less input sources, which always results in larger estimated uncertainties.

Effect of the preprocessing sublayer

From our preprocessing sublayer we now apply the bias estimation module, the map-based outlier handling module, and the module to model autocorrelated errors to the input data. We briefly report on the results of the bias estimation module. Figure 7.36 shows the resulting histogram of the position errors of the two global input sources. It clearly leads to a closer distribution of the u-blox receiver's position error around zero. The statistics provided in Table 7.22 confirm this. Especially the mean lateral error is reduced from previously $0.83\,\mathrm{m}$ to $0.21\,\mathrm{m}$. The longitudinal error improves most noticeably in the MAE from $0.72\,\mathrm{m}$ to $0.38\,\mathrm{m}$. We conclude that the bias estimation on the u-blox data is successful.

Pose fusion performance with preprocessed input data

We plug the pose sources, the preprocessing sublayer, and the core estimator together. Using the PGF with the same parameters as in the experiment in Section 7.5.1, we study its fusion result regarding position error, uncertainty coverage, and timing. Figure 7.37 provides the resulting histograms on the position error. The corresponding statistics are

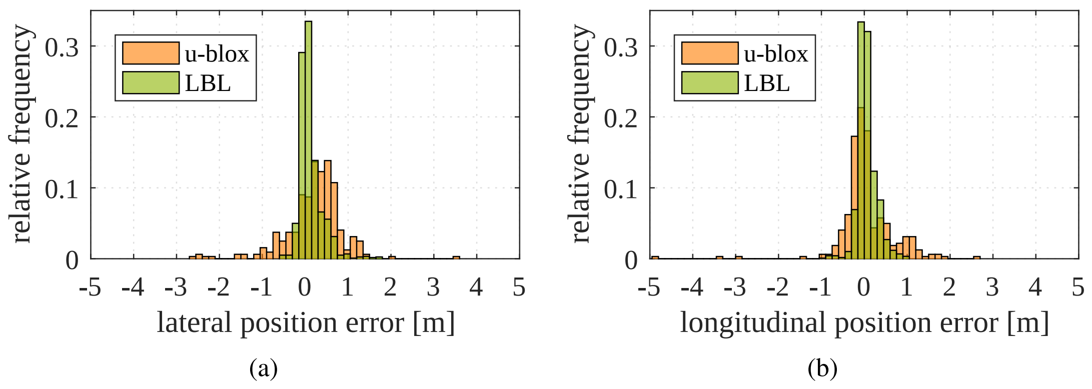


Figure 7.36.: Histogram of the position error of the two global pose sources with bias estimation.

source	direction	mean $[\mathrm{m}]$	median $[\mathrm{m}]$	MAE $[\mathrm{m}]$	RMS $[\mathrm{m}]$	standard deviation $[\mathrm{m}]$
u-blox	lateral	0.21	0.29	0.52	0.69	0.66
	longitudinal	0.06	-0.02	0.38	0.63	0.63
LBL	lateral	0.12	0.05	0.18	0.29	0.26
	longitudinal	0.06	0.03	0.15	0.22	0.22

Table 7.22.: Statistics of the position error of the input sources after applying bias estimation.

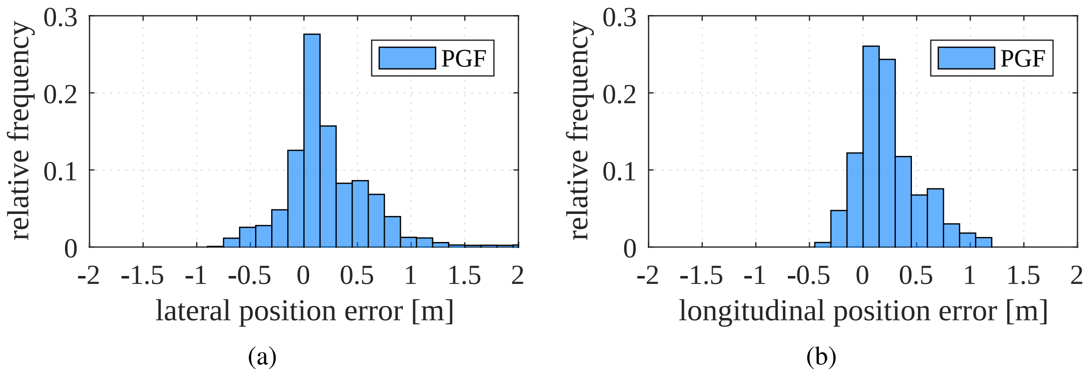


Figure 7.37.: Histogram of the position error of the PGF with preprocessed input data.

error component	mean $[\mathrm{m}]$	median $[\mathrm{m}]$	MAE $[\mathrm{m}]$	RMS $[\mathrm{m}]$	standard deviation $[\mathrm{m}]$
lateral	0.23	0.14	0.33	0.49	0.43
longitudinal	0.24	0.18	0.28	0.37	0.29
Euclidean	0.50	0.45	0.50	0.61	0.36

Table 7.23.: Statistics of the position error of the PGF with preprocessed input data.

given in Table 7.23. Interestingly, the position error stays virtually the same compared to when not using the preprocessing sublayer. Why is that? In the following we look into the details of this test run and analyze some situations more specifically to work out the answers.

The main reason why the position error does not improve here is the interplay of the u-blox receiver and the core estimator. We have seen that the bias estimation indeed performs well and improves the position error of the u-blox receiver. Evidently, it does not alter any of LBL's data. Within the core estimator it weights the global pose constraints with their information matrices. In this experiment the information matrices of the pose estimates from the LBL are on average over a factor of 6.6 larger than those from the u-blox receiver. Additionally, the LBL's output frequency of $5\,\mathrm{Hz}$ is clearly faster than the u-blox receiver's with $1\,\mathrm{Hz}$. As a consequence, the data from the LBL largely dominates the estimation algorithm. Practically, we are only fusing a single global pose source with and without the preprocessing sublayer.

Let us look into two route sections of the test run in more detail. Figure 7.38 provides

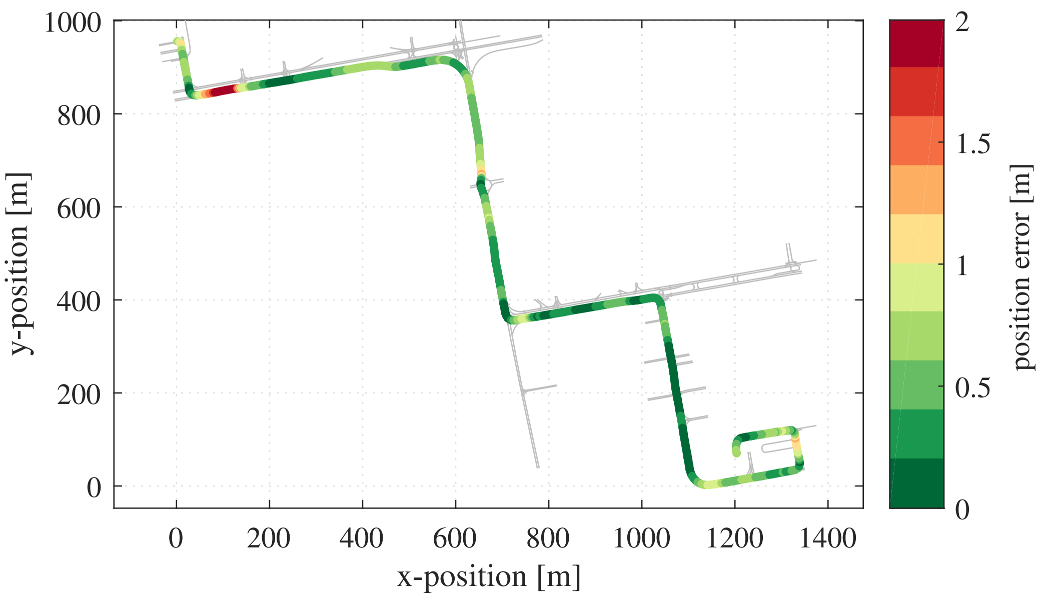


Figure 7.38.: Trajectory of the test run. The color encodes the position error of the PGF.

an overview of the route. The trajectory is color-coded with the position error of the PGF. The gray thin lines are the center lines of the underlying DLM. The start of the trajectory is in the upper left corner. The first section of interest is the red portion in the upper left corner. The second one is during the following right turn, which leads onto a bridge.

To get a better visual impression of the first section, we provide with Figure 7.39 a zoomed-in view. Starting in the top left, both the LBL and the PGF estimate the vehicle pose with little error. Also, the LBL frequently delivers pose estimates. After the vehicle has taken the bend, the LBL's pose estimates are quickly off compared to the reference trajectory. In the same time it also provides few pose estimates because its underlying algorithm needs more computational resources to find a better solution. Once it has found one, it starts outputting at regular intervals again. In comparison, the PGF's sliding window is comprised for quite some time of a lot of observed nodes from before the bend. Using odometry constraints it constructs the graph further in driving direction. In this combination this leads to the reddish colored parts of the trajectory. The PGF corrects its estimated pose only after the new observed nodes from the LBL outweigh the old ones. We call this a *low-pass filter* behavior.

This behavior is what constitutes the core of the PGF. With the odometry information

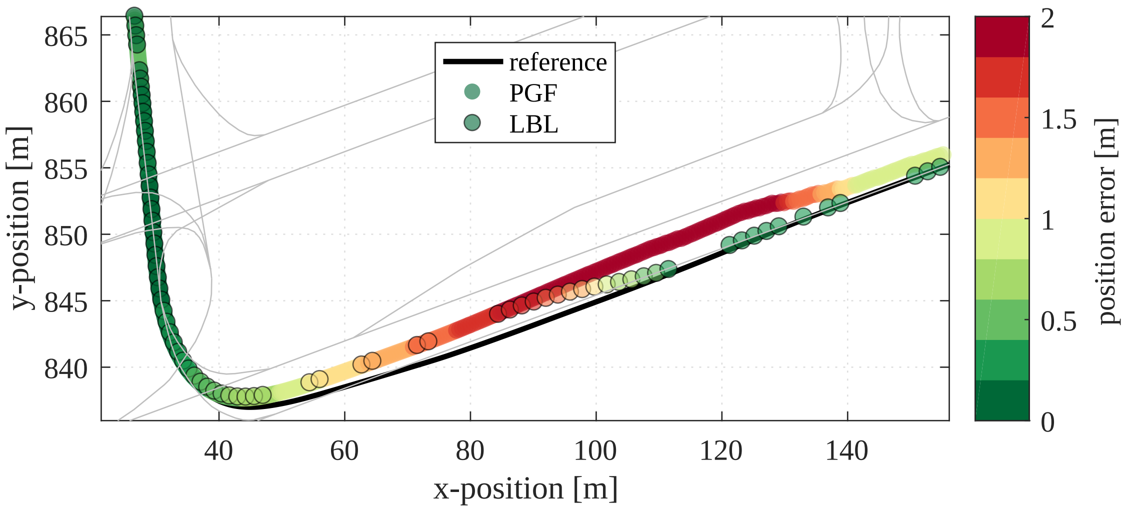
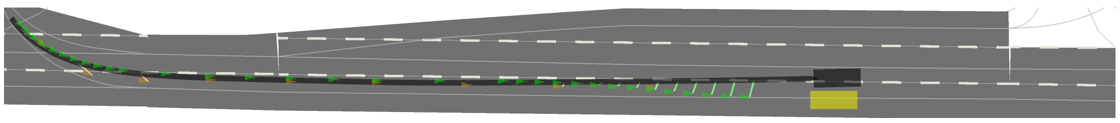


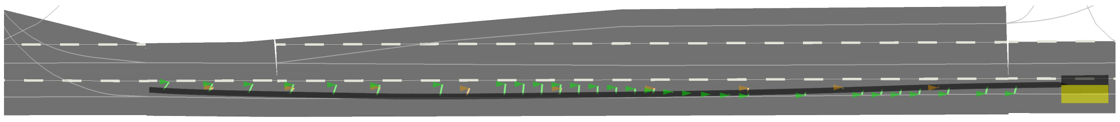
Figure 7.39.: Zoom in on a critical route section.

being rather certain the main task of the PGF is to fit the most recent trajectory through all observed nodes within the sliding window. In the current route section the observed nodes are split up into two rather distinct hypothesis as to where the vehicle recently went. The upper hypothesis leads into the reddish trajectory of what the PGF estimated. The lower hypothesis starts around an x-position of roughly $120\,\mathrm{m}$ and is close to the reference trajectory. As long as the core estimator's sliding window is comprised of mostly observed nodes from the upper hypothesis it will fit the trajectory through that. As soon as the sliding window removes those observed nodes and is dominated by observed nodes from the lower hypothesis the core estimator starts transitioning towards it. We provide two screenshots of our PGF implementation to illustrate this. Figure 7.40a shows the chain pose graph over the sliding window right before the point of transition. Most of the observed nodes in the left part of the sliding window coincide well with the estimated pose of the core estimator for that point, which is shown as dark gray line. The position error is clearly visible as the dark gray rectangle does not overlap with the yellow rectangle at all. In Figure 7.40b we see the pose graph during the transition between the two hypotheses[2]. Old observed nodes at the beginning of the graph are removed and new ones at the front are added. The fusion result starts to better represent the true vehicle pose.

[2]Keep in mind that the covariance matrices of the pose estimates strongly influence the estimation. For visual clarity, these are not shown here.



(a) The pose graph is in good alignment with the left observed nodes. However, they are translated compared to the true recent trajectory.



(b) More observed nodes at the front start to gain importance for the pose graph. This results in a correction of the estimated pose.

Figure 7.40.: Screenshots of the visualization of the PGF within a couple of seconds. The pose graph transitions between the upper (see (a)) and lower hypothesis (see (b)). The hidden nodes are depicted as dark gray line, the observed nodes as green (LBL) and orange (u-blox) triangles, the estimated vehicle pose as dark gray rectangle, and the reference pose as yellow rectangle.

This situation occurs whenever the PGF's input consists of mainly a single global pose source which starts quickly transitioning between two hypothesis for whatever reason. At first, the core estimator considers the new information as outliers. After all, the majority of information at the tail of the sliding window is somewhat in accordance with the old hypothesis. After a while, the new information at the head makes up most of the sliding window. As a result, the optimization fits the estimated trajectory through those observed nodes. Generally, it takes about half of the sliding window to be filled with constraints from a new hypothesis until the core estimator transitions towards it[3]. In other words, the transition time is generally speaking equal to roughly $\frac{1}{2}M\mathit{\Delta}t$.

So, is this low-pass filter behavior unwanted? No; new information, that could well be outliers, should not all of the sudden drastically change the fusion result. This smoothing behavior is the foundation of the pose fusion algorithm. In general, all fusion algorithms will have to decide in such difficult cases when to adapt to the new information. Sometimes, custom-made gating procedures or similar techniques are employed. For a sliding window pose graph, this behavior comes naturally. However, the low-pass behavior admittedly proves to be problematic in the situation at hand. To counterbalance

[3]Here, we assume a constant input frequency of the pose sources, that there is either a single pose source or that all sources behave in the same way, and that their covariance estimates are all within the same order of magnitude.

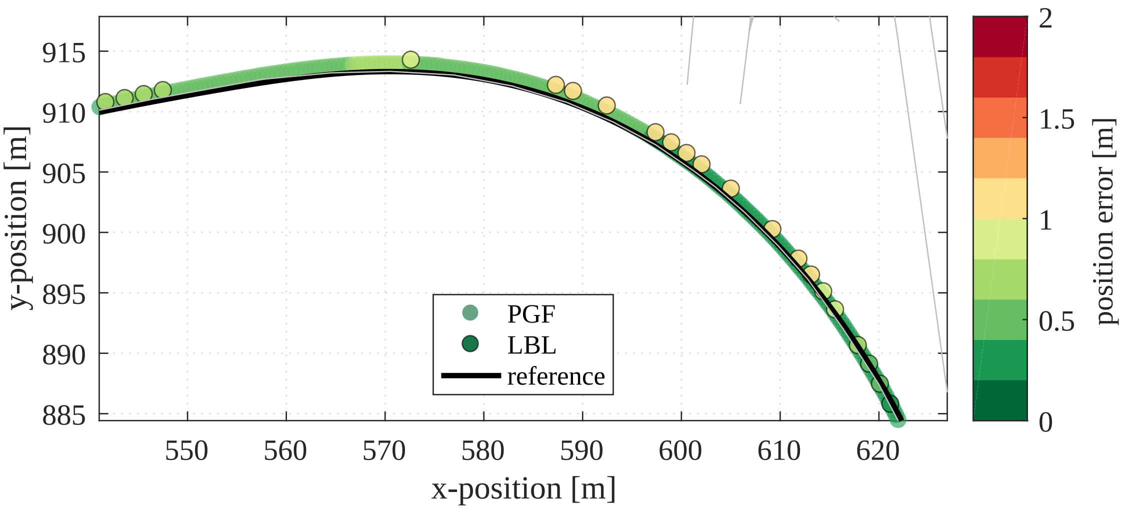


Figure 7.41.: Zoom in on the second route section of interest. The LBL provides irregular pose estimates with a high position error (orange dots). The PGF (greenish line) stays close to the reference trajectory (black line).

this impression, we analyze the second situation in which the opposite happens.

The second route section is located further to the right of the first one and takes place during a right-hand bend onto a bridge. Figure 7.41 shows a zoom onto that route section. The intervals between the pose estimates of the LBL increase and become irregular. Also, their position error temporarily increases significantly. We can see this in the visualization because the dots become orange. Soon after, their position error decreases and the normal behavior is restored. During the entire time span the position quality of the PGF stays more or less constant. The PGF is not influenced by the few successive outliers. The core estimator improves the position quality over the input source in this situation. The low-pass filter behavior of the PGF prevents a too quick adaptation to the new information. Without explicit outlier detection or handling it correctly distinguishes between correct and incorrect pose estimates.

Bringing both examples together, we have seen in the first one that the sliding window can adapt too slowly to new information. The second one demonstrates that it might also be good to not adapt too quickly to new and contradicting information. The desired behavior depends on whether this new information reflects the true trajectory in a better way or not. Often, this can only be answered in hindsight. Options to resolve such conflicts at runtime are to include more input sources with different information.

direction	1σ [%]	2σ [%]	3σ [%]
lateral	53.44	71.84	81.22
longitudinal	56.64	75.00	86.59

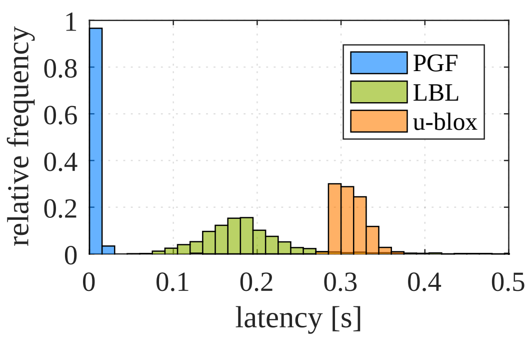
Table 7.24.: Uncertainty coverage of the PGF with preprocessed input data. The estimated uncertainty reflects the actual error significantly better.

Is using the prior node a remedy to this conflict? In some situations perhaps yes, in others likely not. Marginalizing out all older information and using it as prior knowledge hugely improves the output whenever that prior knowledge accurately reflects the truth. If the pose fusion was in a good state, it will stay longer within this and adapt slower to new mismatching information. However, if it was in a bad state, it will also stay longer in this state. This is the effect of temporal smoothing. It leads to the same characteristics as what we called low-pass behavior above. A true remedy to this situation is using more pose sources. Their information can implicitly help to resolve mismatches and decide between conflicting hypothesis.

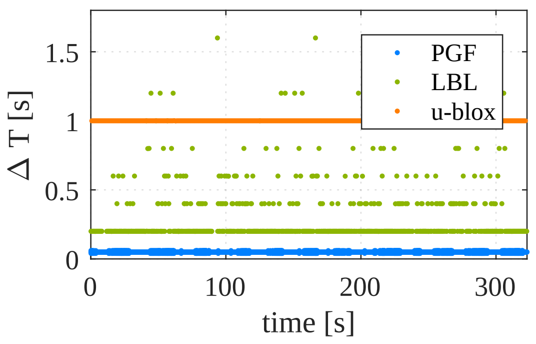
Let us briefly look at other effects of the LBL's behavior. It is common that the LBL provides few estimates in difficult situations. This explains its rather low availability at $73.33\,\%$. The PGF, however, is bound to deliver because the controller of the automated vehicle relies on the current pose. The PGF achieves an availability of $100\,\%$. The statistics of the position errors should be read with this in mind: the position error of the LBL is diluted by only providing pose estimates in good situations at the cost of a lower availability.

We have seen that omitting the preprocessing sublayer leads to an overconfident estimator. Now that we have made use of the preprocessing sublayer we reanalyze its uncertainty coverage. Table 7.24 presents the results. Compared to the previous results given in Table 7.21 we see a strong increase in the uncertainty coverage. We attribute most of this positive effect to our modeling of autocorrelated errors.

Next, we examine the timing of the pose sources and the pose fusion to prove our claims that the PGF handles different and time-varying input frequencies, deals with different latencies of input poses, and provides recent pose estimates at a near-constant output frequency. For these claims we present the plots in Figure 7.42. In Figure 7.42a the distribution of the latencies of the pose estimates of the PGF, the LBL, and the



(a) Distribution of the latencies of the pose estimates per pose source.



(b) Evolution of the time difference $\varDelta T$ between two pose estimates.

Figure 7.42.: Analysis of the timing behavior of the pose sources. The PGF provides recent estimates at a constant output frequency.

u-blox receiver is shown. We recall that the latency of a pose estimate is the difference between the time when it becomes available and the time for which it is valid. The pose estimates of the LBL usually have a latency between $100\,\mathrm{ms}$ to $300\,\mathrm{ms}$, those of the u-blox receiver usually more than $300\,\mathrm{ms}$. In contrast, the pose estimates of the PGF are very recent. The maximum latency is only $20\,\mathrm{ms}$. This is due to the recent odometry data and the core estimator's close attention to its timing behavior. The second plot in Figure 7.42b displays the evolution of $\varDelta T$ per pose source over time. We define $\varDelta T$ as the time difference between two data samples of a pose source. Its examination offers us valuable clues about frequency variations or data dropouts of the corresponding data stream. We observe that the LBL provides samples at irregular intervals. Its base computation cycle is $200\,\mathrm{ms}$ long and it sometimes needs multiple cycles for a computation. The u-blox receiver exhibits a constant output frequency. The PGF brings them both together and outputs on average every $50\,\mathrm{ms}$ a pose estimate. Small fluctuations occur due to minor variations in the computation time c.

In summary, this experiment has offered us valuable insight into some characteristics of the pose fusion. We have seen that if the preprocessing sublayer is not able to model, transform, or reduce unmodeled systematic errors, such as the odometry error, they lead to a degradation of the accuracy.

For systematic offsets of global pose sources we have the bias estimation module. It proved to be successful for the pose estimates from the u-blox receiver. However, in contrast to the experiment in Section 7.5.1 this success did not have significant impact on the outcome of the pose fusion. This is because a strong pose source with low covariance matrices such as the LBL is able to predominate a weaker pose source such as the u-blox receiver. We effectively fused a single global with a single local pose source.

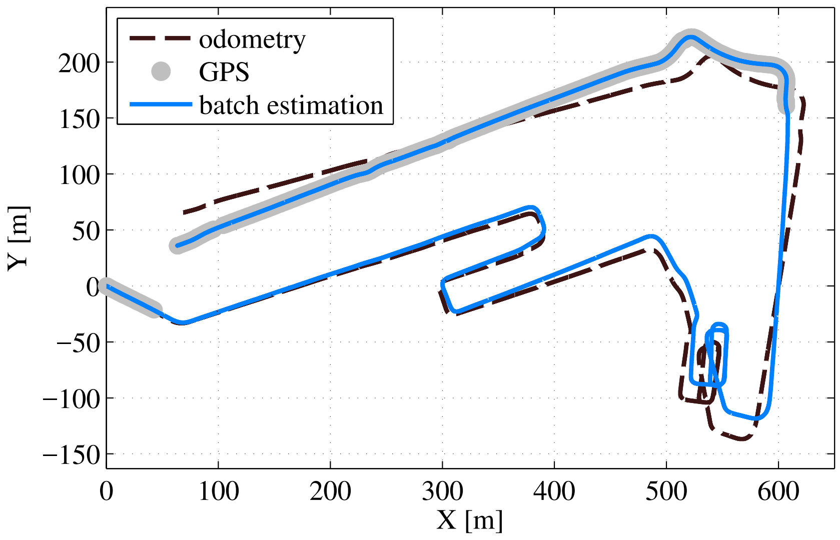
In this combination interesting effects arise. We have seen the low-pass filter behavior of the PGF and its consequences. On the one hand, it has negative aspects like adapting slowly to newer and potentially better information. On the other hand, it has positive effects like not reacting too quickly to consecutive outliers.

Furthermore, we have shown that the PGF handles different and time-varying input frequencies, deals with different latencies of the input poses, provides recent pose estimates at a near-constant output frequency, and is highly available.

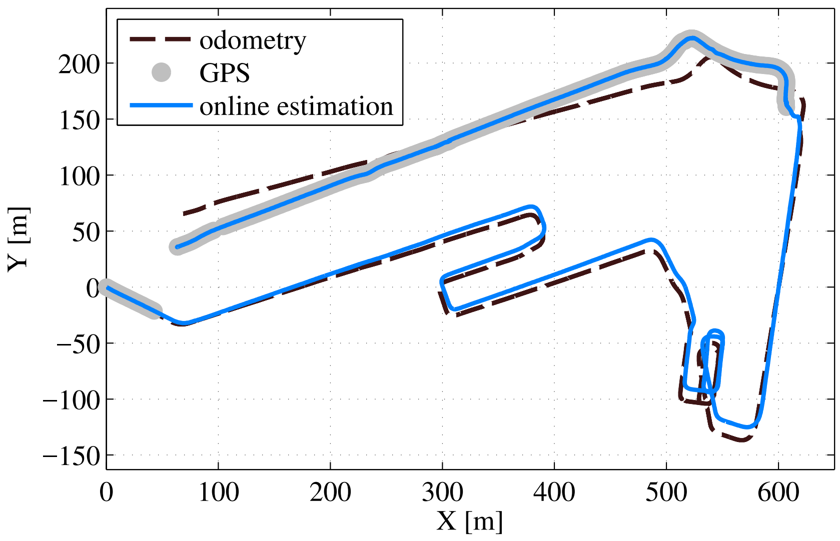
7.5.3. Parking garage

In this section we present an experiment in a parking garage. The vehicle enters a parking garage, traverses it, and exits it at another location. This illustrates our claim that the pose fusion allows the vehicle to transition between different scenarios, where different pose sources are active in each scenario. Here, GPS and odometry data are available outside of the parking garage. Only odometry data is available inside of it. In this part, the pose fusion relies on dead reckoning. The experiment therefore shows that our approach can handle substantial GPS dropouts.

We perform both offline batch estimation and online sliding window smoothing. Offline batch estimation is achieved by setting the number of hidden nodes M to infinity so that the sliding window encompasses all time steps. To obtain the dataset, we recorded roughly $15\,\mathrm{min}$ of odometry and GPS data directly before entering an underground car parking, while driving through it, and at the exit. We filter the valid GPS measurements by requiring at least five satellites to be seen. This results in missing GPS data for the entire transit through the car parking. Additionally, it took a while after exiting it before reliable GPS measurements could be acquired. Figure 7.43 shows the availability of GPS and the accumulated drift in the odometry readings, which amounts to approximately $30\,\mathrm{m}$ and $3.2°$. The figure also shows the optimized global trajectory, once computed offline (Figure 7.43a) and once in an online fashion (Figure 7.43b). The



(a) Our offline batch estimation provides the fully optimized trajectory.



(b) The overall shape of the trajectory of the online estimation is qualitatively comparable to the offline estimation. The most noticeable difference is the transition towards the GPS measurements once they become available again.

Figure 7.43.: We recorded GPS and odometry data while driving through an underground car parking. GPS is only available before entering and shortly after exiting. Both the online and offline estimation are able to handle substantial GPS dropouts.

online sliding window smoothing is able to perform dead reckoning. The associated part of its estimated trajectory shows the typical dead reckoning effect of accumulated error. However, it approximates the trajectory of the batch estimation after GPS data becomes available again. It is therefore able to cope with substantial GPS dropouts.

8. Conclusion and future work

In this thesis we have studied the problem of estimating the pose of an automated vehicle with different pose sources, also called generic pose fusion. We briefly summarize our findings here and discuss limitations and possible future work.

8.1. Summary and discussion

The key question of this work is how to design a generic pose fusion approach. We propose a layered architecture with a core estimator and a set of preprocessing modules for this. The core estimator is the backbone of our approach. It fits the recent trajectory as measured by the odometry through the set of supporting points as measured by the global pose sources. For this we construct a pose graph over a sliding window of pose estimates. Optimizing this graph leads to the maximum likelihood (ML) solution over the joint probability of local and global pose estimates in the current window. We have shown that this converges to the online batch estimate for increasing sizes of the sliding window. Also, adapting this window size allows us to design the core estimator as a resource-adaptive algorithm.

The core estimator makes general assumptions about the nature of the error of the input data. The preprocessing techniques account for error modes that are specific to certain input sources. They transform the input data in a suitable way. We propose four preprocessing modules: bias estimation, map-based outlier handling, treatment of cross-correlated errors between pose sources, and modeling of autocorrelated errors. In its practical application, these modules have proven to be of different value to us. The bias estimation directly influences the accuracy of the estimation. We found it simple to implement and effective in its application. The map-based outlier handling is a straightforward way of reducing the influence of outlier constraints. Naturally, how much it affects the pose fusion is dependent on the dataset and the input sources. Robust cost functions may be a viable alternative. The application of the cross-correlated error treatment turned out to be the least useful of the preprocessing techniques. First, it is unclear when to use it. On the one hand, analyzing data from two different pose sources can show random correlations that do not warrant the existence of cross-correlated errors. On the other hand, pose sources that we strongly expect from a theoretical perspective to exhibit cross-correlated errors turn out to not show significant correlations, for example because of very different implementations. This makes it unclear when to use this method. Secondly, applying it to pose estimates with covariance matrices in different orders of magnitude simply leads to throwing away the larger one. This common effect is easy to achieve without the computational complexity of Covariance Intersection (CI). The fourth preprocessing module is the modeling of autocorrelated errors. While its theoretical derivation is rather complex, it is straightforward to implement as according scaling of information matrices. Its application leads to improved uncertainty estimates.

One focus of this thesis is to extend the expressiveness of pose graphs. To this end, we derived the prior node, which is essentially the graphical model of the effect of marginalization of chain pose graphs. We also derived how to model factors that represent autocorrelated errors of local and global pose constraints. Even without implementing them explicitly, it is useful to have a graphical model of how they affect the graph to understand their influence.

We have experimentally verified that our system fulfills the requirements that we formulated in the beginning. More precisely, we have shown it to be highly available, recent, working online, resource-adaptive, extendable to six degrees of freedom, compatible with certain preprocessing techniques, accurate, and able to provide reasonable estimates of its uncertainty. These are important prerequisites to apply it to multiple pose sources and vehicles. We demonstrated this capability by letting it run on multiple vehicles and with different configurations and inputs.

8.2. Limitations and future work

The strongest advantage of our approach is also its biggest weakness: its generic nature. While it allows us to plug in any kind of pose source, we suffer from the lack of knowledge of their underlying algorithms. This prevents a tight integration, thorough modeling, and rigorous error propagation. In the future, it would be beneficial to study hybrid approaches between generic and specific fusion. Richer stochastic models could be developed for certain pose sources while maintaining the generic treatment of the remaining sources. This could be supported by additional preprocessing modules aimed at transforming the input data in suitable ways.

We tested the pose fusion on automated vehicles. It would be interesting to see how much of it can be directly transferred to other use cases. Thinking of light-weight drones, for example, there might be different or additional requirements. Usually, they have limited computing power. This could lead to the need for a closer integration of the graph construction with the graph optimization. We rely on a plain configuration of a widespread graph optimization framework because it is fast enough for our use. If this is not the case in different use cases, then it might be beneficial to encode knowledge about sliding window chain pose graphs directly in the graph optimization. This could prevent the rebuilding of most of the problem from scratch in every iteration and instead keep most of the structure intact.

The concepts in this thesis build up on pose graph optimization. This methodology is currently the state-of-the-art technique for solving Simultaneous Localization and Mapping (SLAM) problems. Therefore, it is a straightforward future strand of research to study how to bring together pose fusion with SLAM. Ideally, this leads to combining the benefits of a tight integration of SLAM features with the loosely coupled fusion of pose sources. In this way, the pose fusion would not only merge other pose source's data, but become a pose source on its own.

Lastly, our conclusion from this thesis is that our key concepts results in a simple, yet general, take on the generic pose fusion problem. In research environments, its makes sense to develop such a system as derived in this thesis. It allows the easy integration of new pose sources. Also, robots can be equipped with different pose sources but they could share the same pose fusion system. This increases the reuse of components, reduces the effort to implement new specific fusion systems, and overall renders the pose fusion future-proof. It is a different situation when an application or a product shall be developed based on a specific set of pose sources. In this case, it makes sense to adopt some key concepts of this thesis and to aim for a hybrid approach as outlined above. This ensures that the system is extensible in the future as well as making full use of the knowledge of the specific pose sources.

List of Figures

List of Tables

A. Automotive software environment

This chapter gives an overview of the automotive software framework in which the concepts presented in this thesis are implemented.

ADTF (Schabenberger, 2007) is an automotive middleware developed by Elektrobit[1]. It is commercially available for Linux and Microsoft Windows operating systems. In the automotive industry it is widely-used by major automotive manufacturing companies, such as Volkswagen, Daimler, and their suppliers.

As a middleware it focuses on coupling software components and opening up the possibility to exchange data between them. These software components are called *filters* and they are generally written in C++. Technically, they are shared libraries that are loaded by the ADTF runtime process. Figure A.1 depicts two filters that receive and transmit binary data packets (so-called *MediaSamples*) via their pins. Developers that are familiar with Robot Operating System (ROS) will quickly find similarities between nodes in ROS and filters in ADTF. For developers that are more familiar with MATLAB Simulink the concept of Simulink's blocks lends itself to comparison with filters.

ADTF offers a graphical configuration editor that allows the coupling and parametriza-

[1]`https://www.elektrobit.com/products/eb-assist/adtf/`

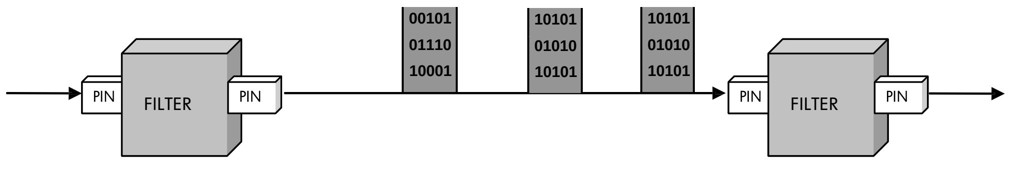


Figure A.1.: ADTF streaming architecture. Filters exchange binary data over pins. Source: EB Assist ADTF Support (2015, Fig. 1.4).

tion of filters. Filters can be placed and connected via drag-and-drop as this creates a directed graph. An edge in this graph defines the coupling between two filters and therefore the data-flow between software components. An individual coupling of filters is called a configuration and can be stored and loaded. Figure A.2 shows a screenshot of an example ADTF development environment with two filters in the configuration editor. The progress bar at the top indicates that ADTF is currently playing back recorded data. This is opposed to the online mode in which the data streams represent the current sensor readings in a vehicle. As many middlewares, ADTF supports these two modes, playback and online mode, to enable the prototyping of new software modules with existing test data. The recorded data is saved in so-called .dat-files (similar to .bag-files under ROS) and can be played back at the workstation. The same filters can be used in both modes.

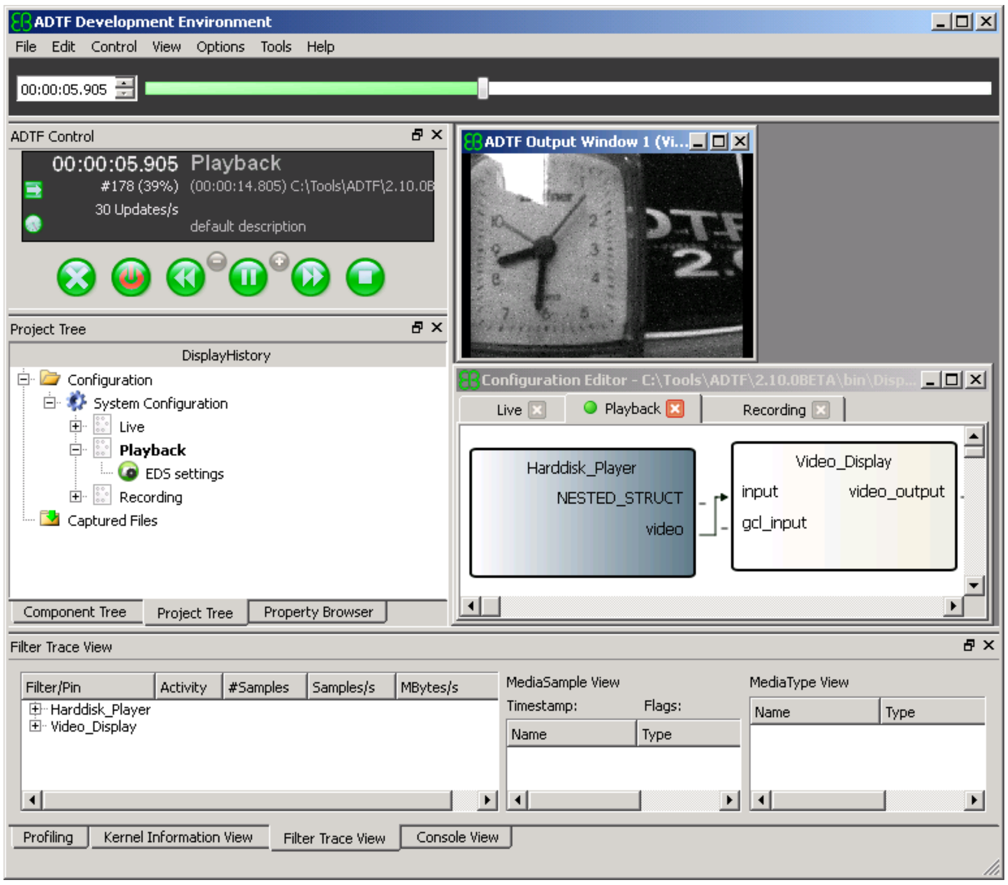


Figure A.2.: ADTF development environment. In this case, it consists of the configuration editor with two filters (middle right), the control panel (upper left), an output window (upper right), and several other elements. Source: EB Assist ADTF Support (2015, Fig. 2.9).

Abbreviations

ABS	anti-lock braking system 145
ADAS	advanced driver assistance system 4, 6, 32
ADTF	Automotive Data and Time-Triggered Framework 76, 211–213
AR(1)	autoregressive model of order 1 66, 67, 115–118, 120, 124, 125, 128, 132, 133, 136, 169–172, 183, 185
AWGN	additive white Gaussian noise 18, 22, 26, 46, 74, 115, 117, 118, 120, 123, 124, 127, 170, 174, 175, 188
BDS	BeiDou Navigation Satellite System 30
CAN bus	controller area network bus 145
CEP	circular error probable 141, 142
CI	Covariance Intersection 9, 25, 61–66, 69, 70, 111–114, 133–135, 168, 169, 204
CPU	Central Processing Unit 96, 97, 157–159
CRLB	Cramér-Rao Lower Bound 67–69, 133
DBN	Dynamic Bayesian Network 18, 48, 49, 52
DCS	Dynamic Covariance Scaling 24, 25
DGPS	Differential Global Positioning System 32
DLM	Detailed Lane Model 35–37, 107, 108, 164, 183, 194
ECU	electronic control unit 145

EDF	empirical distribution function 151, 153–155, 177, 189, 190
EIF	Extended Information Filter 17
EKF	Extended Kalman filter 14–19, 22, 145, 147
ENU	East North Up 29
ESP	electronic stability program 33, 145
FI	Fisher Information 67–70, 133, 134
g2o	General Graph Optimization 50, 80, 87
GLONASS	Global Navigation Satellite System 30, 138, 141, 142
GNSS	Global Navigation Satellite System 1, 3–5, 30–32, 71, 115, 138, 141, 142, 147
GPS	Global Positioning System 17, 22, 23, 25, 26, 30–33, 71, 72, 78, 102, 111, 138–142, 147, 158, 160, 200–202
GSM	Global System for Mobile Communications 30
GTSAM	Georgia Tech Smoothing and Mapping library 50
IEKF	Iterated Extended Kalman filter 8, 14–17, 19, 79
i.i.d.	independent and identically distributed 170
IMU	inertial measurement unit 2, 5, 14, 17, 22, 31, 33, 71, 72, 138, 141, 142, 145, 147, 188
INS	inertial navigation system 17, 33
iSAM	Incremental Smoothing and Mapping 50
LBL	landmark-based localization 146, 164–166, 173–176, 178, 179, 181–184, 188, 189, 192–194, 196–200
lidar	light detection and ranging 3, 33, 34, 138–140, 143, 146
LiLoc	lidar localization 146, 165, 166, 173–176, 178, 179, 181–184

LSDF	localization sensor data fusion 147, 173–179, 181–184
MAE	mean absolute error 161, 166, 172, 175, 180–182, 184–186, 189, 191–193
ML	maximum likelihood 8, 13, 18, 46, 53, 79, 203
MRF	Markov random field 18, 48, 49, 51, 59
NDS	Navigation Data Standard 34, 35
NLLSQ	nonlinear least squares 13, 18–20, 23, 26, 37–39, 42, 44, 46, 47, 52–55, 59, 60, 67, 79–81, 88, 98, 111, 115, 116, 118, 120, 123–125, 128, 132, 133, 135, 136, 149, 166
NLOS	non-line-of-sight 25, 31
OEMStar	NovAtel FlexPak-G2 OEMStar 138, 149, 158, 160, 161, 165, 166, 173–176, 178, 179, 181–184, 187
PACF	partial autocorrelation function 175–178, 183
PC	Personal Computer 137, 138
PDF	probability density function 65, 67, 68
PGF	PoseGraphFusion 76–78, 95, 96, 149, 150, 152, 153, 156–158, 161, 164–167, 169–172, 178–181, 183, 185–187, 190, 191, 193–200
PID	proportional-integral-derivative 97, 157–159, 179
PPP	Precise Point Positioning 32
QZSS	Quasi-Zenith Satellite System 141, 142
radar	radio detection and ranging 22, 33, 34, 137, 146
RFID	radio-frequency identification 33
RMS	root mean square 138, 142, 166, 175, 180, 182, 186, 189, 191–193

ROS	Robot Operating System 211, 212
RRR	Realizing, Reversing, Recovering 24, 25
RSSI	Received Signal Strength Indication 30
RTK	Real Time Kinematics 32, 142
SC	Switchable Constraints 24, 25
SEIF	Sparse Extended Information Filter 17
SIFT	scale-invariant feature transform 146
SLAM	Simultaneous Localization and Mapping 9, 13, 14, 19, 23, 25, 32, 49, 50, 54, 59, 80, 98, 115, 146, 205
sSBA	Sparse Sparse Bundle Adjustment 50
SWF	Sliding Window Filter 15, 16, 19, 20, 79
u-blox	u-blox M8 ADR 138, 141, 149, 187–189, 191–193, 196, 199, 200
UKF	Unscented Kalman Filter 18
UTM	Universal Transverse Mercator 28, 29, 32
WGM	Weighted Geometric Mean 25, 65, 66

Bibliography

Gabriel Agamennoni, Paul Furgale, and Roland Siegwart. Self-tuning M-estimators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635, 2015.

Pratik Agarwal and Edwin Olson. Variable reordering strategies for SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3844–3850, 2012.

Pratik Agarwal, Gian Diego 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)*, pages 62–69, 2013.

Pratik Agarwal, Wolfram Burgard, and Cyrill Stachniss. Helmert's and Bowie's geodetic mapping methods and their relation to graph-based SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3619–3625. IEEE, 2014a.

Pratik Agarwal, Wolfram Burgard, and Cyrill Stachniss. A survey of geodetic approaches to mapping and the relationship to graph-based SLAM. *IEEE Robotics & Automation Magazine*, 21(3):63–80, 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)*, pages 3111–3118, 2015.

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Sietz, and Rick Szeliski. Building Rome in a day. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 72–79, 2009.

Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17 (4):886–905, 1996.

Applanix. POS LV, 2017. URL `https://www.applanix.com/products/poslv.htm`. Accessed November 21, 2018.

Michael Baer, Mohamed Essayed Bouzouraa, Christopher Demiral, Ulrich Hofmann, Stefan Gies, and Klaus Diepold. EgoMaster: A central ego motion estimation for driver assist systems. *Proceedings of the IEEE International Conference on Control and Automation (ICCA)*, pages 1708–1715, 2009.

Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

Tim Bailey, Simon J. Julier, and Gabriel Agamennoni. On conservative fusion of information with unknown non-gaussian dependence. In *Proceedings of the International Conference on Information Fusion (FUSION)*, pages 1876–1883, 2012.

Yaakov Bar-Shalom. Update with out-of-sequence measurements in tracking: Exact solution. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):769–778, 2002.

Timothy D. Barfoot. *State Estimation For Robotics*. Cambridge University Press, 2017.

Bradley Bell and Frederick Cathey. The Iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993.

Jose-Luis Blanco. A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga, 2010.

Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 979–984, 1989.

Peter J. Brockwell, Rainer Dahlhaus, and A. Alexandre Trindade. Modified burg algorithms for multivariate subset autoregression. *Statistica Sinica*, pages 197–213, 2005.

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)*, pages 5748–5755, 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)*, pages 854–860, 2014.

Nicholas Carlevaris-Bianco, Michael Kaess, and Ryan M. Eustice. Generic node removal for factor-graph SLAM. *IEEE Transactions on Robotics*, 30(6):1371–1385, 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)*, pages 1926–1931, 2016.

Subhash Challa, Robin J. Evans, Xuezhi Wang, and Jonathan Legg. A fixed-lag smoothing solution to out-of-sequence information fusion problems. *Communications in Information and Systems (CIS)*, 2(4):325–348, 2002.

Lingji Chen, Pablo O. Arambel, and Raman K. Mehra. Estimation under unknown correlation: Covariance Intersection revisited. *IEEE Transactions on Automatic Control*, 47(11):1879–1882, 2002.

Han Pang 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)*, pages 46–53, 2013.

Han-Pang Chiu, Xun S. Zhou, Luca Carlone, Frank Dellaert, Supun Samarasekera, and Rakesh Kumar. Constrained optimal selection for multi-sensor robot navigation using plug-and-play factor graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 663–670, 2014.

Chi Kin Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

Donald Cochrane and Guy H. Orcutt. Application of least squares regression to relationships containing auto-correlated error terms. *Journal of the American Statistical Association*, 44(245):32–61, 1949.

Davide A. Cucci and Matteo Matteucci. A flexible framework for mobile robot pose estimation and multi-sensor self-calibration. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 361–368, 2013.

Igor Cvišić and Ivan Petrović. Stereo odometry based on careful feature selection and tracking. In *Proceedings of the European Conference of Mobile Robots (ECMR)*, 2015.

Michael Darms and Hermann Winner. A modular system architecture for sensor data processing of ADAS applications. In *IEEE Intelligent Vehicles Symposium*, pages 729–734, 2005.

Thomas L. Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 524–529. American Association for Artificial Intelligence, 1988.

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):1181–1203, 2006.

Tue-Cuong Dong-Si and Anastasios I. Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5655–5662, 2011.

Gregory Dudek and Michael Jenkin. Inertial sensing, GPS and odometry. In *Springer Handbook of Robotics*, chapter 29, pages 737–752. Springer International Publishing, 2016.

James Durbin and Geoffrey S. Watson. Testing for serial correlation in least squares regression. I. *Biometrika*, 37(3/4):409–428, 1950.

Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping (SLAM): Part I the essential algorithms. *IEEE Robotics & Automation Magazine*, 13(2):99–108, 2006.

EB Assist ADTF Support. *User's Manual*. Elektrobit Automotive GmbH, 2015. Version 2.12.0.

Christian Eling. *Entwicklung einer direkten Georeferenzierungseinheit zur Positions- und Orientierungsbestimmung leichter UAVs in Echtzeit*. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn, 2016.

Christian Eling, Lasse Klingbeil, and Heiner Kuhlmann. Real-time single-frequency GPS/MEMS-IMU attitude determination of lightweight UAVs. *Sensors*, 15(10): 26212–26235, 2015.

Wilfried Elmenreich. *Sensor Fusion in Time-Triggered Systems*. Ph.D. dissertation, Technical University of Vienna, 2002.

Jakob Engel, Jürgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1449–1456, 2013.

Ryan M. Eustice, Hanumant Singh, and John J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Transactions on Robotics*, 27(6):1100–1114, 2006a.

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):1223–1242, 2006b.

Georgios Floros, Benito Van Der Zander, and Bastian Leibe. OpenStreetSLAM: Global vehicle localization using OpenStreetMaps. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1054–1059, 2013.

John Folkesson and Henrik Christensen. Graphical SLAM – a self-correcting map. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 383–390, 2004.

John Folkesson and Henrik I. Christensen. Closing the loop with graphical SLAM. *IEEE Transactions on Robotics*, 23(4):731–741, 2007.

Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2014.

Wolfgang Förstner. Graphical models in geodesy and photogrammetry. *Journal of Photogrammetry, Remote Sensing, and Geoinformation Science (PFG)*, 2013(4):255–267, 2013.

Wolfgang Förstner and Bernhard P. Wrobel. Graphical models in geodesy and photogrammetry. In *Photogrammetric Computer Vision*, chapter 4, pages 75–185. Springer International Publishing, 2016.

David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002. ISBN 0130851981.

Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012.

Udo Frese and Gerd Hirzinger. Simultaneous localization and mapping – a discussion. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Reasoning with Uncertainty in Robotics (IJCAI)*, pages 17–26, 2001.

Udo Frese and Lutz Schröder. Theorie der Sensorfusion. Lecture notes, 2007. URL `http://www.informatik.uni-bremen.de/agebv/de/VeranstaltungTDS06`.

Lukas Fröhlich. *Improving a Multi-Sensor Data Fusion for Localization of Automated Vehicles*. M.Sc. thesis, Swiss Federal Institute of Technology Zurich (ETH), 2017.

Daniel Gabay. Minimizing a differentiable function over a differential manifold. *Journal of Optimization Theory and Applications*, 37(2):177–219, 1982.

Carl-Friedrich Gauss. *Theoria Combinationis Observationum Erroribus Minimis Obnoxia (Theory of the Combination of Observations Least Subject to Error).* Henricus Dieterich, 1823.

Audrey Giremus, Arnaud Doucet, Anne-Christine Escher, and Jean-Yves Tourneret. Nonlinear filtering approaches for INS/GPS integration. In *Proceedings of the European Signal Processing Conference*, pages 873–876, 2004.

Matteo Golfarelli, Dario Maio, and Stefano Rizzi. Elastic correction of dead-reckoning errors in map building. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 905–911, 1998.

Irwin R. Goodman, Ronald P. Mahler, and Hung T. Nguyen. *Mathematics of data fusion*, volume 37. Springer Science & Business Media, 2013.

Mohinder S. Grewal, Lawrence R. Weill, and Angus P. Andrews. *Global positioning systems, inertial navigation, and integration.* John Wiley & Sons, 2007.

Ralph Grewe, Andree Hohm, Stefan Hegemann, Stefan Lueke, and Hermann Winner. Towards a generic and efficient environment model for ADAS. In *IEEE Intelligent Vehicles Symposium*, pages 316–321, 2012.

Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, pages 31–43, 2010a.

Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, Udo Frese, and Christoph Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 273–278, 2010b.

Paul D. Groves, Ziyi Jiang, Morten Rudi, and Philip Strode. A portfolio approach to NLOS and multipath mitigation in dense urban areas. In *Proceedings of the International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS)*, pages 3231–3247, 2013.

Jens-Steffen Gutmann and Kurt Konolige. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, 2000.

Karl-Ludwig Haken. *Grundlagen der Kraftfahrzeugtechnik*. Carl Hanser Verlag GmbH Co KG, Munich, Germany, 2013.

Anders Hald. *A history of parametric statistical inference from Bernoulli to Fisher, 1713–1935*. Springer-Verlag New York, 2008. ISBN 978-0-387-46408-4.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

Toni Heidenreich, Jens Spehr, and Christoph Stiller. LaneSLAM – simultaneous pose and lane estimation using maps with lane-level accuracy. In *IEEE Transactions on Intelligent Transportation Systems*, pages 2512–2517, 2015.

Christoph Hertzberg. *A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds*. M.Sc. 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):57–77, 2013.

Timo Hinzmann, Thomas Schneider, Marcin Dymczyk, Andreas Schaffner, Simon Lynen, Roland Siegwart, and Igor Gilitschenski. Monocular visual-inertial SLAM for fixed-wing UAVs using sliding window based nonlinear optimization. In *International Symposium on Visual Computing*, pages 569–581, 2016.

Jeroen Hol. *Sensor fusion and calibration of inertial sensors, vision, ultra-wideband and GPS*. Ph.D. dissertation, Linköping University, Sweden, 2011.

Sebastian Houben, Marcel Neuhausen, Matthias Michael, Robert Kesten, Florian Mickler, and Florian Schuller. Park marking-based vehicle self-localization with a fisheye topview system. *Journal of Real-Time Image Processing*, pages 1–16, 2015.

Guoquan Huang, Michael Kaess, and John J. Leonard. Consistent sparsification for graph optimization. In *Proceedings of the European Conference of Mobile Robots (ECMR)*, pages 150–157, 2013.

Shoudong Huang and Gamini Dissanayake. A critique of current developments in simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 13(5):1–13, 2016.

Vadim Indelman, Stephen Williams, Michael Kaess, and Frank Dellaert. Factor graph based incremental smoothing in inertial navigation systems. In *Proceedings of the International Conference on Information Fusion (FUSION)*, pages 2154–2161, 2012.

International Organization for Standardization. ISO 8855:2011 road vehicles – vehicle dynamics and road-holding ability – vocabulary, 2011. URL `http://www.iso.org/iso/catalogue_detail.htm?csnumber=51180`.

Kichun Jo, Keonyup Chu, and Myoungho Sunwoo. GPS-bias correction for precise localization of autonomous vehicles. In *IEEE Intelligent Vehicles Symposium*, pages 636–641, 2013.

Simon Julier. The stability of covariance inflation methods for SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2749–2754, 2003.

Simon J. Julier. Fusion of dependent information in Posegraphs. In *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 504–509, 2012.

Simon J. Julier and Jeffrey K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the American Control Conference (ACC)*, pages 2369–2373, 1997.

Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

Simon J. Julier and Jeffrey K. Uhlmann. Using covariance intersection for SLAM. *Robotics and Autonomous Systems*, 55(1):3–20, 2007.

Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, pages 1198–1210, 2009.

Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research*, 31(2):216–235, 2012.

Sören Kammel, Julius Ziegler, Benjamin Pitzer, Moritz Werling, Tobias Gindele, Daniel Jagzent, Joachim Schröder, Michael Thuy, Matthias Goebl, and Felix von Hundelshausen. Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):615–639, 2008.

Alonzo Kelly. *Mobile Robotics: Mathematics, Models, and Methods*. Cambridge University Press, 2013.

Bahador Khaleghi, Alaa Khamis, Fakhreddine O. Karray, and Saiedeh N. Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion*, 14(1): 28–44, 2013.

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.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

Kurt Konolige. Large-scale map-making. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 457–463, 2004.

Kurt Konolige. Sparse sparse bundle adjustment. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 102–1, 2010.

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)*, pages 865–871, 2011.

Frank Kschischang, Brendan Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

Vladimír Kubelka, Lorenz Oswald, François Pomerleau, Francis Colas, Tomáš Svoboda, and Michal Reinstein. Robust data fusion of multimodal sensory information for mobile robots. *Journal of Field Robotics*, 32(4):447–473, 2015.

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)*, pages 1–8, 2014.

Heiner Kuhlmann. Kalman-filtering with coloured measurement noise for deformation analysis. In *Proceedings of the FIG Symposium on Deformation Measurements*, 2003.

Rainer Kümmerle. *State Estimation and Optimization for Mobile Robot Navigation*. Ph.D. dissertation, University of Freiburg, 2013.

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)*, pages 3607–3613, 2011.

Jean Laneurit, Roland Chapuis, and Frédéric Chausse. Accurate vehicle positioning on a numerical map. *International Journal of Control, Automation, and Systems*, 3(1): 15–31, 2005.

Thomas Dall Larsen, Nils A. Andersen, Ole Ravn, and Niels Kjølstad Poulsen. Incorporation of time delayed measurements in a discrete-time Kalman filter. In *Proceedings of the IEEE Conference on Decision and Control*, volume 4, pages 3972–3977, 1998.

Henning Lategahn, Markus Schreiber, Julius Ziegler, and Christoph Stiller. Urban localization with camera and inertial measurement unit. In *IEEE Intelligent Vehicles Symposium*, pages 719–724, 2013.

Yasir Latif, César Cadena, and José Neira. Robust loop closing over time for pose graph SLAM. *International Journal of Robotics Research*, 32(14):1611–1626, 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)*, pages 2683–2690, 2014.

Adrien Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, 34(3):314–334, 2015.

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.

Wangyan Li, Zidong Wang, Guoliang Wei, Lifeng Ma, Jun Hu, and Derui Ding. A survey on multisensor fusion and consensus filtering for sensor networks. *Discrete Dynamics in Nature and Society*, 2015, 2015.

Steven Lovegrove, Andrew J. Davison, and Javier Ibañez Guzmán. Accurate visual odometry from a rear parking camera. In *IEEE Intelligent Vehicles Symposium*, pages 788–793, 2011.

Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.

Xiaoguang Luo. *GPS stochastic modelling: signal quality measures and ARMA processes*. Ph.D. dissertation, Karlsruhe Institute of Technology, 2013.

Simon Lynen, Markus W. Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to MAV navigation.

In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3923–3929, 2013.

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)*, pages 62–69, 2015.

Gangadharrao S. Maddala and Kajal Lahiri. *Introduction to Econometrics*, volume 2. New York: Macmillan, 1992.

Ronald P. S. Mahler. Optimal/robust distributed data fusion: a unified approach. In *Proceedings of the Conference Signal Processing, Sensor Fusion, and Target Recognition (SPIE)*, volume 4052, pages 128–138, 2000.

Norman Mattern, Robin Schubert, and Gerd Wanielik. High-accurate vehicle localization using digital maps and coherency images. In *IEEE Intelligent Vehicles Symposium*, pages 462–469, 2010.

Peter S. Maybeck. *Stochastic models, estimation, and control*, volume 3. Academic press, 1982.

Mladen Mazuran, Gian Diego Tipaldi, Luciano Spinello, and Wolfram Burgard. Nonlinear graph sparsification for SLAM. In *Proceedings of the Robotics: Science and Systems Conference (RSS)*, pages 1–8, 2014.

Mladen Mazuran, Wolfram Burgard, and Gian Diego Tipaldi. Nonlinear factor recovery for long-term SLAM. *International Journal of Robotics Research*, 35(1-3):50–72, 2016.

Christian Merfels. *Large-scale Probabilistic Feature Mapping and Tracking for Autonomous Driving*. M.Sc. thesis, Technical University of Darmstadt, 2014.

Christian Merfels. Kompensation von Fehlern in Absolut-Positionsdaten bei der Schätzung der Eigenposition. Patent application at Deutsches Patent- und Markenamt, Germany, October 2016. DE 10 2016 220 593.5.

Christian Merfels and Moritz Schack. Fusion von Positionsdaten mittels Posen-Graph. Patent application at Deutsches Patent- und Markenamt, Germany, October 2015. DE 10 2015 219 577.5.

Christian Merfels and Moritz Schack. Marginalisieren eines Posen-Graphen. Patent application at Deutsches Patent- und Markenamt, Germany, March 2016. DE 10 2016 205 193.8.

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)*, pages 3116–3123, 2016.

Christian Merfels and Cyrill Stachniss. Sensor fusion for self-localization of automated vehicles. *Journal of Photogrammetry, Remote Sensing, and Geoinformation Science (PFG)*, 85(2):113–126, 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 ITS)*, 2016. ISSN: 2191-8287.

Christian Merfels, Lukas Fröhlich, and Bernd Rech. Verfahren zur Datenfusion eines Datensatzes, entsprechende Recheneinheit und Fahrzeug welches mit einer entsprechenden Recheneinheit ausgestattet ist sowie Computerprogramm. Patent application at Deutsches Patent- und Markenamt, Germany, April 2017a. 10 2017 108 130.5.

Christian Merfels, Lukas Fröhlich, Bernd Rech, Thilo Schaper, Niklas Koch, and Daniel Wilbers. Verfahren, Vorrichtung und computerlesbares Speichermedium mit Instruktionen zur Schätzung einer Pose eines Kraftfahrzeugs. Patent application at Deutsches Patent- und Markenamt, Germany, April 2017b. DE 10 2017 108 107.0.

Marco Miani. Spherical and cartesian grids, 2009. URL `http://www.texample.net/tikz/examples/spherical-and-cartesian-grids`. Accessed November 21, 2018.

Isaac Miller, Mark Campbell, and Dan Huttenlocher. Map-aided localization in sparse global positioning system environments using vision and particle filtering. *Journal of Field Robotics*, 28(5):619–643, 2011.

Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, 2002.

Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, pages 335–348, 2014.

Michael Munz. *Generisches Sensorfusionsframework zur gleichzeitigen Zustands-und Existenzschätzung für die Fahrzeugumfelderkennung*. Ph.D. dissertation, Universität Ulm, 2011.

Michael Munz, Klaus Dietmayer, and Mirko Mählisch. Generalized fusion of heterogeneous sensor measurements for multi target tracking. In *Proceedings of the International Conference on Information Fusion (FUSION)*, 2010a.

Michael Munz, Mirko Mählisch, Jürgen Dickmann, and Klaus Dietmayer. Probabilistic modeling of sensor properties in generic fusion systems for modern driver assistance systems. In *IEEE Intelligent Vehicles Symposium*, pages 760–765, 2010b.

Michael Munz, Mirko Mahlisch, and Klaus Dietmayer. Generic centralized multi sensor data fusion based on probabilistic sensor and environment models for driver assistance systems. *IEEE Intelligent Transportation Systems Magazine*, 2(1):6–17, 2010c.

Robin R. Murphy. Dempster-shafer theory for sensor fusion in autonomous mobile robots. *IEEE Transactions on Robotics and Automation*, 14(2):197–206, 1998.

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):1406–1433, 2009.

Johannes Niebuhr and Gerhard Lindner. *Physikalische Messtechnik mit Sensoren*. Oldenbourg Industrieverlag München, 2002.

Wolfgang Niemeier. *Ausgleichsrechnung – Statistische Auswertemethoden*. de Gruyter, second edition, 2008. ISBN 978-3110190557.

David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–659, 2005.

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)*, pages 1862–1867. IEEE, 2015.

NovAtel Inc. FlexPak-G2 with OEMStar, 2015. URL `https://www.novatel.com/products/gnss-receivers/enclosures/flexpak-g2-with-oemstar`. Accessed November 21, 2018.

Edwin Olson. *Robust and Efficient Robotic Mapping*. Ph.D. dissertation, Massachusetts Institute Of Technology, 2008.

Edwin Olson. Real-time correlative scan matching. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4387–4393, 2009.

Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *International Journal of Robotics Research*, 32(7):826–840, 2013.

Edwin Olson, John Leonard, and Seth J. Teller. Fast iterative alignment of pose graphs with poor initial estimates. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2262–2269, 2006.

Oxford Technical Solutions Ltd. RT3003, 2016. URL `http://www.oxts.com/product/rt3003/`. Accessed November 21, 2018.

Tim Pfeifer, Peter Weissig, Sven Lange, and Peter Protzel. Robust factor graph optimization–a comparison for sensor fusion applications. In *Proceedings of the International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2016.

Mareike Ploog. *Uncertainty Recovery of a Pose Fusion for Self-Localization of Automated Vehicles*. M.Sc. thesis, University of Hannover, 2017.

Hua Qin, Yang Peng, and Wensheng Zhang. Vehicles on RFID: Error-cognitive vehicle localization in GPS-less environments. *IEEE Transactions on Vehicular Technology*, PP(99), 08 2017.

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)*, pages 2486–2493, 2007.

Matthias Rapp, Michael Barjenbruch, Klaus Dietmayer, and Markus Hahn. A fast probabilistic ego-motion estimation framework for radar. In *Proceedings of the European Conference of Mobile Robots (ECMR)*, 2015.

Denise Ratasich, Bernhard Frömel, Oliver Höftberger, and Radu Grosu. Generic sensor fusion package for ROS. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 286–291, 2015.

Marc Reinhardt, Benjamin Noack, and Uwe D. Hanebeck. Closed-form optimization of covariance intersection for low-dimensional matrices. In *Proceedings of the International Conference on Information Fusion (FUSION)*, pages 1891–1896, 2012.

Matt Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Decentralized sensor fusion with distributed particle filters. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 493–500, 2002.

Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry : Part I: The first 30 years and fundamentals. *IEEE Robotics & Automation Magazine*, 18(4):80–92, 2011. ISSN 10709932.

Roland Schabenberger. ADTF: Framework for driver assistance and safety systems. In *Proceedings of the International Conference on Electronic Systems for Vehicles*, pages 701–710. VDI-Gesellschaft Fahrzeug- und Verkehrstechnik, 2007.

Tobias Scheide, Martin Escher, Hans-Georg Büsing, and Peter Hecker. Integere Fahrzeugortung für zukünftige Fahrerassistenzsysteme. In *Proceedings of the Posi-*

tioning and Navigation for Intelligent Transportation Systems Conference (POSNAV ITS), 2011.

Bruno Scherzinger. Precise robust positioning with inertial/GPS RTK. In *Proceedings of the International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS)*, pages 115–162, 2000.

Steven Schlegel, Nico Korn, and Gerik Scheuermann. On the interpolation of data with normally distributed uncertainty for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2305–2314, 2012.

Alexander Schlichting and Claus Brenner. Localization using automotive laser scanners and local pattern matching. In *IEEE Intelligent Vehicles Symposium*, pages 414–419, 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)*, pages 4784–4791, 2016a.

Johannes Schneider, Cyrill Stachniss, and Wolfgang Förstner. On the accuracy of dense fisheye stereo. *IEEE Robotics and Automation Letters (RA-L)*, 1(1):227–234, 2016b.

K. P. Schwarz, Ming Wei, and M. Van Gelderen. Aided versus embedded – a comparison of two approaches to GPS/INS integration. In *Proceedings of the IEEE Position, Location and Navigation Symposium (PLANS)*, pages 314–322, 1994.

Gabe Sibley. A sliding window filter for SLAM. Technical report, University of Southern California, 2006.

Gabe Sibley. *Long Range Stereo Data-Fusion from Moving Platforms*. Ph.D. dissertation, University of Southern California, 2007.

Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010.

Joris Sijs and Mircea Lazar. State fusion with unknown correlation: Ellipsoidal intersection. *Automatica*, 48(8):1874–1878, 2012.

Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

Cyrill Stachniss. *Exploration and mapping with mobile robots*. Ph.D. dissertation, University of Freiburg, 2006.

Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. Simultaneous localization and mapping. In *Springer Handbook of Robotics*, chapter 46, pages 1153–1176. Springer International Publishing, 2016.

Nico Steinhardt and Stefan Leinen. Data fusion for precise localization. In *Handbook of Driver Assistance Systems*, pages 605–646. Springer International Publishing, 2015.

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.

Stephen M. Stigler. The epic story of maximum likelihood. *Statistical Science*, 22(4): 598–620, 2007.

Hauke Strasdat, J. Montiel, and Andrew J. Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

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)*, pages 1879–1884, 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)*, pages 5198–5203, 2013.

Zui Tao, Philippe Bonnifait, Vincent Frémont, and Javier Ibañez-Guzman. Mapping and localization using GPS, lane markings and proprioceptive sensors. In *Proceedings of* *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 406–412, 2013.

Cédric Tessier, Christophe Cariou, Christophe Debain, Frederic Chausse, Roland Chapuis, and Christophe Rousset. A real-time, multi-sensor architecture for fusion of delayed observations: application to vehicle localization. In *IEEE Transactions on Intelligent Transportation Systems*, pages 1316–1321, 2006.

Sebastian Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the new Millennium*, pages 1–35. Morgan Kaufmann, 2002.

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):403–429, 2006.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.

William F. Tinney and John W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE*, 55(11):1801–1809, 1967.

Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. *Proceedings of the International Workshop on Vision Algorithms: Vision and Practice*, pages 298–372, 2000.

u-blox. NEO-M8L series—u-blox M8 ADR modules with 3D sensors, 2017. URL `https://www.u-blox.com/en/product/neo-m8l-series`. Accessed November 21, 2018.

Jeffrey K. Uhlmann. *Dynamic Map Building and Localization for Autonomous Vehicles*. Ph.D. dissertation, University of Oxford, 1995.

Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, and Jason Ziglar. Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

Velodyne LiDAR Inc. PUCK VLP-16, 2017. URL `http://velodynelidar.com/vlp-16.html`. Accessed November 21, 2018.

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)*, pages 886–893, 2011.

Volkswagen AG. Level 5 Urban Car Mobility System, 2017. URL `http://www.discover-future-mobility.com/download/`. Accessed November 21, 2018.

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)*, pages 4511–4516, 2016.

Tianmiao Wang, Chaolei Wang, Jianhong Liang, Yang Chen, and Yicheng Zhang. Vision-aided inertial navigation for small unmanned aerial vehicles in GPS-denied environments. *International Journal of Advanced Robotic Systems*, 10, 2013.

Erik Ward and John Folkesson. Vehicle localization with low cost radar sensors. In *IEEE Intelligent Vehicles Symposium*, pages 864–870, 2016.

Stephan Weiss, Markus W. Achtelik, Margarita Chli, and Roland Siegwart. Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 31–38, 2012.

Klaudius Werber, Michael Barjenbruch, Jens Klappstein, Jürgen Dickmann, and Christian Waldschmidt. RoughCough - a new image registration method for radar based vehicle self-localization. In *Proceedings of the International Conference on Information Fusion (FUSION)*, pages 1533–1541, 2015.

Antje Westenberger. *Simultane Zustands-und Existenzschätzung mit chronologisch ungeordneten Sensordaten für die Fahrzeugumfelderfassung*. Ph.D. dissertation, Universität Ulm, 2015.

Sardha Wijesoma, Kwang Wee Lee, and Javier Ibanez Guzman. On the observability of path constrained vehicle localisation. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1513–1518, 2006.

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)*, pages 176–183, 2014.

Hua Xue, Hongzi Zhu, Siyuan Cao, Shan Chang, and Jian Cao. UPS: Combatting urban vehicle localization with cellular-aware trajectories. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, 2016.

George Udny Yule. On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London (Series A)*, 226:267–298, 1927.

Farzaneh Zangeneh-Nejad, Ali R. Amiri-Simkooei, Mohammad A. Sharifi, and Jamal Asgari. On the realistic stochastic model of GPS observables: Implementation and performance. In *Proceedings of the International Conference on Sensors & Models in Remote Sensing & Photogrammetry*, 2015.

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)*, pages 2174–2181, 2015.

Julius Ziegler, Henning Lategahn, Markus Schreiber, Christoph G. Keller, Carsten Knöppel, Jochen Hipp, Martin Haueis, and Christoph Stiller. Video based localization for BERTHA. In *IEEE Intelligent Vehicles Symposium*, pages 1231–1238, 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):53–60, 2016.