Localization in Urban Environments – A Hybrid Interval-Probabilistic Method

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur

(abgekürzt Dr.-Ing.)

genehmigte Dissertation

von Herrn M. Sc. Aaronkumar Ehambram
geboren am 27.01.1996 in Rinteln

2023

1. Referent: Prof. Dr.-Ing. Bernardo Wagner
2. Referent: Prof. Dr.-Ing. Luc Jaulin (Brest, Frankreich)

Tag der Promotion: 22. August 2023

Acknowledgements

This dissertation was written as part of my work as a research assistant in the Department of Real-Time Systems (RTS) at the Institute of Systems Engineering at the Leibniz Universität Hannover. Furthermore, I was a research associate of the Research Training Group i.c.sens [RTG 2159] funded by the Deutsche Forschergemeinschaft (DFG).

First and foremost, I would like to express my sincere gratitude to Prof. Dr.-Ing. Bernardo Wagner for his support, excellent guidance, and constant motivation. Even when I was not very confident about the progress of my doctoral research, he always encouraged me and provided numerous alternatives and insights that motivated me to carry out my research project.

A heartfelt thank you also goes to Prof. Luc Jaulin, who supported this work with extensive technical discussions, new ideas, and enthusiasm. I am also grateful to him for taking on the role of the second examiner for this dissertation. I admire his passion and conviction for intervals and their application in robotics.

I want to thank my colleagues at the RTS and DEI for the wonderful time we spent together. I particularly appreciate the pleasant atmosphere and our sense of teamwork. The table tennis tournaments and console nights were always much fun, and I hope we can maintain this tradition.

The time I spent in Luc's laboratory in Brest will always remain a great memory for me, where I was able to develop professionally and personally. Special thanks go to Simon Rohou, who not only provided research ideas and incentives but also made the stay very enjoyable. Discussions with the members of the Lab-STICC enabled significant scientific exchange, and valuable friendships were formed.

Furthermore, I am thankful for my time as a Research Training Group i.c.sens member, where we fostered close scientific exchange through joint measurement campaigns, hackathons, Jour Fixes, and professor's days.

I am incredibly grateful to the proofreaders of this work, who provided me with valuable feedback for the final version of this dissertation. I want to extend my special thanks and heartfelt gratitude to Volker Wellhöner, Julian Kunze, Jan-Michel Rotter, Sousa Weddige, Raphael Voges, Sven Wagner, Henrik Jürgens, Gia Bao Thieu, Anja Wojcik, Michael Zebrowski, Sarah Ehambram, Rachel Ehambram, and Anika Wojcik.

Lastly, I would like to thank my beloved family and friends for their unwavering support, motivation, and encouragement. We have not always had an easy journey, and without my mother and my siblings, I would not be here writing this dissertation. I am genuinely grateful to have all of you in my life. My deepest gratitude goes to my partner, Anja, who, especially during the stressful phase of my doctoral studies, always untiringly assisted me, showed great understanding, and supported my deep enthusiasm and passion for my work.

Aaronkumar Ehambram

June 2023

Kurzfassung

Mit zunehmender Autonomie von Fahrzeugen ist die Gewährleistung der Sicherheit zu einem vorrangigen Anliegen geworden. Eine grundlegende Aufgabe dabei ist die Lokalisierung, die für einen sicheren Betrieb unerlässlich ist. Zur Quantifizierung der Sicherheitsanforderungen wurde das Konzept der Integrität eingeführt. Die Integrität beschreibt die Fähigkeit des Systems rechtzeitig und korrekt zu warnen, wenn ein sicherer Betrieb nicht mehr gewährleistet werden kann. Um jedoch die Funktionsfähigkeit des Systems abschätzen zu können, muss unter anderem die Unsicherheit der Lokalisierung bewertet werden.

In der Literatur existieren zwei vorherrschende Ansätze – die Wahrscheinlichkeitstheorie und die Mengenzugehörigkeitstheorie, die mathematische Werkzeuge zur Bewertung der Unsicherheit zur Verfügung stellen. Probabilistische Ansätze liefern oft gute Ergebnisse, neigen aber dazu den Fehler zu unterschätzen. Bei mengentheoretischen Ansätzen hingegen wird die Unsicherheit zuverlässig bewertet, der Fehler jedoch tendenziell überschätzt. Während die Unterschätzung des Fehlers zu gefährlichen Systemausfällen ohne Vorwarnung führen kann, machen zu pessimistische Schätzungen das System unbrauchbar.

Das Ziel dieser Dissertation ist es, die symbiotische Beziehung zwischen mengenbasierten und probabilistischen Lokalisierungsansätzen zu untersuchen und sie zu einem einheitlichen, hybriden Lokalisierungsansatz zur Fehlerabschätzung zu kombinieren. In dieser Arbeit wird eine neue Lokalisierungsmethode vorgestellt – die sogenannte **H**ybrid **P**robabilistic- **a**nd **S**et-Membership-based **Co**arse and **Re**fined (HyPaSCoRe) Lokalisierung. Diese Methode lokalisiert einen Roboter in einer Gebäudekarte in Echtzeit und berücksichtigt zwei Arten der Hybridisierung. Einerseits werden mengenbasierte Ansätze verwendet, um probabilistische Ansätze in ihrem Lösungsbereich einzuschränken und somit die Robustheit zu erhöhen. Andererseits werden probabilistische Ansätze verwendet, um den Pessimismus mengentheoretischer Ansätze zu reduzieren, indem zusätzliche probabilistische Bedingungen hinzugefügt werden.

Die Methode besteht aus drei Modulen: visuelle Odometrie, grobe Lokalisierung und verfeinerte Lokalisierung. Die HyPaSCoRe Lokalisierung konzentriert sich auf die Lokalisierung in städtischen Gebieten, in denen GNSS-Daten ungenau sein können. Das Modul für die visuelle Odometrie berechnet die relative Bewegung des Fahrzeugs. Die grobe Lokalisierung schränkt die Menge der möglichen Posen mit einem mengenbasierten Ansatz ein und erweitert die Schätzung mit einer probabilistischen Methode, um die wahrscheinlichsten Lösungen innerhalb der Menge zu bestimmen. Das Modul für die verfeinerte Lokalisierung präzisiert das Ergebnis, indem es den Pessimismus der mengenbasierten Unsicherheitsschätzung durch weitere probabilistische Bedingungen reduziert.

Die experimentelle Untersuchung zeigt, dass die Integrität der Unsicherheitsabschätzung gewährleistet ist, während präzise Lokalisierungsergebnisse in Echtzeit geliefert werden. Die Einführung dieses neuen hybriden Lokalisierungsansatzes stellt einen Beitrag zur Entwicklung sicherer und zuverlässiger Algorithmen im Kontext des autonomen Fahrens dar.

Schlagworte:
Autonomes Fahren, Lokalisierung in Gebäudekarten, Hybride Interval-Probabilistische Lokalisierung, Mengenbasierte Fehlerabschätzung, Intervallarithmetik, Probabilistische Fehlerabschätzung

Abstract

Ensuring safety has become a paramount concern with the increasing autonomy of vehicles and the advent of autonomous driving. One of the most fundamental tasks of increased autonomy is localization, which is essential for safe operation. To quantify safety requirements, the concept of integrity has been introduced in aviation, based on the ability of the system to provide timely and correct alerts when the safe operation of the systems can no longer be guaranteed. Therefore, it is necessary to assess the localization's uncertainty to determine the system's operability.

In the literature, probability and set-membership theory are two predominant approaches that provide mathematical tools to assess uncertainty. Probabilistic approaches often provide accurate point-valued results but tend to underestimate the uncertainty. Set-membership approaches reliably estimate the uncertainty but can be overly pessimistic, producing inappropriately large uncertainties and no point-valued results. While underestimating the uncertainty can lead to misleading information and dangerous system failure without warnings, overly pessimistic uncertainty estimates render the system inoperative for practical purposes as warnings are fired more often.

This doctoral thesis aims to study the symbiotic relationship between set-membership-based and probabilistic localization approaches and combine them into a unified hybrid localization approach. This approach enables safe operation while not being overly pessimistic regarding the uncertainty estimation. In the scope of this work, a novel **H**ybrid **P**robabilistic- **a**nd **S**et-Membership-based **Co**arse and **Re**fined (HyPaSCoRe) Localization method is introduced. This method localizes a robot in a building map in real-time and considers two types of hybridizations. On the one hand, set-membership approaches are used to robustify and control probabilistic approaches. On the other hand, probabilistic approaches are used to reduce the pessimism of set-membership approaches by augmenting them with further probabilistic constraints.

The method consists of three modules – visual odometry, coarse localization, and refined localization. The HyPaSCoRe Localization uses a stereo camera system, a LiDAR sensor, and GNSS data, focusing on localization in urban canyons where GNSS data can be inaccurate. The visual odometry module computes the relative motion of the vehicle. In contrast, the coarse localization module uses set-membership approaches to narrow down the feasible set of poses and provides the set of most likely poses inside the feasible set using a probabilistic approach. The refined localization module further refines the coarse localization result by reducing the pessimism of the uncertainty estimate by incorporating probabilistic constraints into the set-membership approach.

The experimental evaluation of the HyPaSCoRe Localization shows that it maintains the integrity of the uncertainty estimation while providing accurate, most likely point-valued solutions in real-time. Introducing this new hybrid localization approach contributes to developing safe and reliable algorithms in the context of autonomous driving.

Keywords:
Autonomous Driving, Localization in Building Maps, Hybrid Interval-Probabilistic Localization, Set-Membership-based Uncertainty Models, Interval Analysis, Probabilistic Uncertainty Models

Contents

Acronyms

AMCL	Adaptive Monte Carlo Localization
BPF	Box-Particle Filter
CSP	Constraint Satisfaction Problem
DOF	Degrees Of Freedom
EKF	Extended Kalman Filter
FoV	Field of View
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HyPaSCoRe Localization	Hybrid Probabilistic and Set-Membership-based Coarse and Refined Localization
ICP	Iterative Closest Points
iHT	interval-based Hough Transformation
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LM	Levenberg-Marquardt Optimization
LOD	Level Of Detail
MCL	Monte Carlo Localization
OpenCV	Open Source Computer Vision Library
OSM	OpenStreetMap
PDF	Probability Density Function
PF	Particle Filter
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
RBPF	Rao-Blackwellized Particle Filter
SfM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SIVIA	Set Inversion Via Interval Analysis
SLAM	Simultaneous Localization and Mapping
ToF	Time of Flight
UAV	Unmanned Aerial Vehicle
UTM	Universal Transverse Mercator

Symbols

Interval analysis

x	Scalar value (or an arbitary variable)
$\mathbf{x}$	Vector of values, $\mathbf{x} \in \mathbb{R}^n$
$[x]$	Interval
$[\mathbf{x}]$	Interval vector (or box)
$\underline{x}$	Lower bound of $[x]$
$\overline{x}$	Upper bound of $[x]$
$w([x])$	Width of $[x]$
$\mathsf{mid}([x])$	Midpoint of $[x]$
x^*	Actual (unknown) value of x

Coordinate systems

C_t	General camera coordinate system at time t
I_t	General image pixel coordinate system at time t
$C_{l,t}$	Left camera coordinate system at time t
$I_{l,t}$	Left image pixel coordinate system at time t, image coordinate frame origin is defined in the top left corner
$C_{r,t}$	Right camera coordinate system at time t
$I_{r,t}$	Right image pixel coordinate system at time t
L_t	Laser scanner coordinate system at time t
M	Map coordinate system, fixed in space and cannot change over time

Transformations and poses

${}^A\mathbf{R}_B$	Rotation matrix describing the rotation between coordinate systems A and B
${}^A\boldsymbol{\xi}_B = (\,\varphi_B^A \;\; \theta_B^A \;\; \psi_B^A\,)^\intercal$	Euler angles describing the rotation between coordinate systems A and B
${}^A\varphi_B$	Yaw angle
${}^A\theta_B$	Pitch angle
${}^A\psi_B$	Roll angle

${}^{A}\mathbf{t}_{B}$	Translation vector describing the shift between the coordinate system B described in A
${}^{A}\mathbf{T}_{B} = \begin{pmatrix} {}^{A}\mathbf{R}_{B} & {}^{A}\mathbf{t}_{B} \\ \mathbf{0} & 1 \end{pmatrix}$	Homogeneously extended affine transformation to transform from frame B to A
${}^{A}\hat{\mathbf{T}}_{B} = ({}^{A}\mathbf{t}_{B}^{\intercal} \;\; {}^{A}\boldsymbol{\xi}_{B}^{\intercal})^{\intercal}$	Pose of frame B described in frame A, the parameter representation of ${}^{A}\mathbf{T}_{B}$
${}^{A}\mathcal{T}_{B}$	General set of translations describing the shift between coordinate systems A and B
${}^{A}\Phi_{B}$	General set of rotations between coordinate systems A and B
${}^{A}\mathcal{P}_{B} = \left({}^{A}\mathcal{T}_{B} \;\; {}^{A}\Phi_{B} \right)^{T}$	General set of poses of frame B described in frame A
${}^{A}\mathcal{P}_{B}^{\boxplus}$	Set of poses represented by a subpaving of frame B described in frame A
${}^{A}\mathcal{P}_{B}^{\triangle}$	Set of poses represented by a polygon of the translation and an interval for the rotation of frame B described in frame A

Coordinates

r, α, β	Spherical coordinates consisting of the distance r, polar angle α and azimuthal angle β
$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$	Intrinsic camera projection matrix
${}^{A}\mathbf{p}_{i} = ({}^{A}p_{x,i} \;\; {}^{A}p_{y,i} \;\; {}^{A}p_{z,i})^{\intercal}$	3D coordinates of the point i in the coordinate system A
${}^{I_t}\mathbf{p}_{i} = ({}^{I_t}p_{x,i} \;\; {}^{I_t}p_{y,i})^{\intercal}$	2D image pixel point at time t

Miscellaneous

${}^{M}B_{i} = \{ {}^{M}\mathbf{c}_{1}, \ldots, {}^{M}\mathbf{c}_{n} \}$	Building polygon i described in the map frame M defined by a set of corner points ${}^{M}\mathbf{c}_{j}$
${}^{M}\mathcal{B} = \{ {}^{M}B_{1}, \ldots, {}^{M}B_{n} \}$	Set of buildings described in the map frame M

1 Introduction

In 2021, 2.3 million traffic accidents were registered in Germany – $258\,987$ of them involved human injury. Human error was by far the most frequent cause: $88.0\,\%$ of the accidents were caused by driver misconduct, according to the German Federal Statistical Office. [1, 2]

Due to the increasing maturity of driver assistance systems in modern vehicles, the number of accidents generally decreases, suggesting that more technical assistance during driving improves traffic safety [1]. One goal of increasing vehicle autonomy is to further reduce the number of accidents caused by driver misconduct and thereby increase road safety. The ultimate objective in the field of highly automated driving is still to achieve full autonomy [2]. In our society, the expectation that these systems will have fully deterministic, comprehensible, and safe behavior is very high [3, 4]. Nevertheless, the question remains: Is it possible to achieve this ideal goal from a technical point of view?

Driving a vehicle involves many different tasks. For example, obstacles must be detected, and the vehicle must determine whether it may collide with the objects. Is the detected obstacle just a leaf from a tree, or is it a living creature the vehicle could harm in case of a collision? Object detection is, therefore, one of the critical tasks to be solved. Localization poses another fundamental problem: The vehicle needs to localize itself on a map to plan a trajectory and navigate from the starting point to the destination. Is the vehicle on the road or too close to the sidewalk where potential pedestrians may be at risk? Are the driver assistance functions authorized in the localized area of the map? These are just a few examples of different tasks that combine different research areas where we need to analyze the determinism, comprehensibility, and safety of the sensors and algorithms. Within the scope of this thesis, we will restrict ourselves to the localization problem as a small but fundamental part.

We have already alluded to terms such as safety and reliability. As stated in [5], **safety** is defined as the absence of unacceptable risk. Risk, however, is defined as "[...] a combination of the probability of fault occurrence and the severity of corresponding consequences" [4, 6]. Fault occurrence is conceptually tied to the **reliability**: The higher the fault occurrence, the lower the reliability. As a result, safety considers both the probability of a fault and its consequences. A system can only be called safe if faults with severe consequences only occur with an acceptably low probability [4]. To quantify the requirements for safety, the term **integrity** has to be introduced.

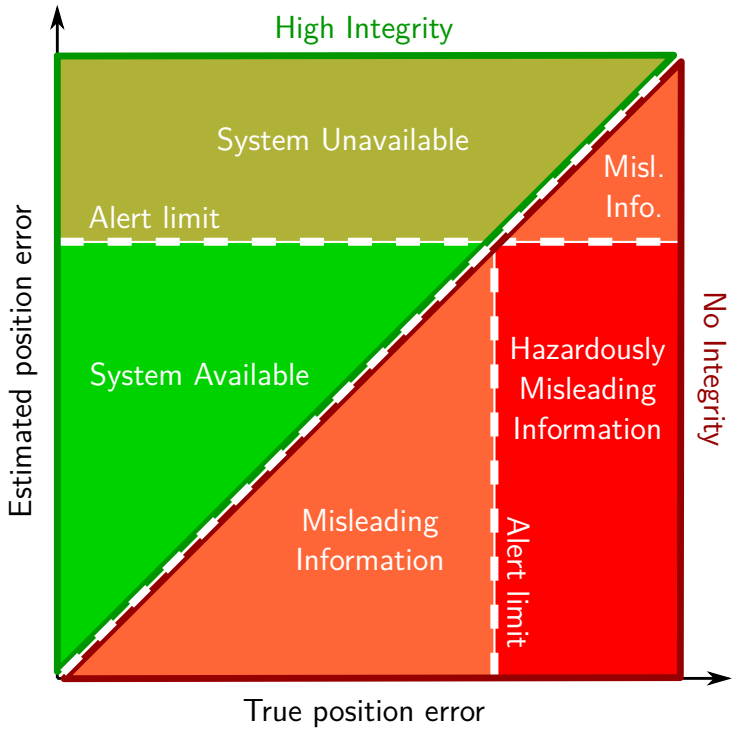


Figure 1.1: Integrity levels in the Stanford-ESA Integrity Diagram based on [4, 7]. The horizontal axis measures the true error, and the vertical axis the estimated error. The alert limit defines the five regions that correspond to the integrity levels.

1.1 Integrity

The integrity of a system is a "[...] measure of the trust that can be placed in the correctness of the information supplied by the total system [...]" [4]. Furthermore, the integrity of a system includes the ability to provide timely and valid alerts to the user in case that the system must not be used for the intended operation [4]. Hence, regarding localization, the vehicle must be able to warn the passengers if the localization uncertainty exceeds an alert limit. The integrity of a system is usually classified into integrity levels, which are visualized in the Stanford-ESA Integrity Diagram in Figure 1.1 based on [7].

While the horizontal axis defines the true position error, which is usually unknown during regular operation, the vertical axis represents the estimated upper bound of the true position error provided by the localization system. An ideal localization system would provide perfect estimates of the localization error where all pairs of true and estimated errors lie in the diagonal line. Operating the system in the upper left triangle with green areas allows the safe operation of the localization system because the estimated error is greater than or equal to the true error. Consequently, the system has high integrity as the results are trustworthy. However, operation in the lower right triangle with red regions defines unsafe operation because the localization system underestimates the true error. The system has no integrity since the uncertainty assessment is not reliable. [4, 7]

The alert limit defines the estimated error limit above which the system will alert the user that the vehicle cannot be operated safely. In this case, either the passengers must take action or the vehicle must stop. As shown in the Stanford diagram in Figure 1.1, the alert limit divides

the diagram into five sections. At best, the localization system stays in the green region. At worst, its estimates are in the lower right triangle. The operation is acceptable as long as the true error is below the alert limit. However, suppose the true error exceeds the alert limit while the estimated error is still below the alert limit. In that case, the localization system enters the *Hazardously Misleading Information* region. The operation becomes dangerous because the user is not warned as the estimated error is too optimistic even though the true error has exceeded the alert limit.

As a consequence, estimating the error is a vital part of measuring the integrity and, therefore, the safety of a system. The estimation of the possible error is known as risk assessment. How can we compute the risk? This question leads us to the problem statement and the research questions this thesis addresses.

1.2 Problem Statement and Research Questions

In the literature, different methods exist to assess the risk of a localization estimate. The central idea is to propagate the uncertainty of the measurements to the estimate. In contrast to other works [8, 9], we use the term **uncertainty** as a measure of fuzziness of measurements and states. Consequently, we will use the terms risk and uncertainty synonymously throughout this work, assuming events to occur with measurable probability.

The most commonly used uncertainty models are probabilistic models that consider the uncertainty by a probability distribution. The uncertainty of the localization estimate is assessed by propagating the error distribution from the sensor readings to the localization estimate. However, the first problem that arises with probabilistic approaches is the selection of the appropriate probability distribution. By appealing to the Central Limit Theorem [10], and due to convenient mathematical properties of the Gaussian function, most probabilistic localization approaches assume pure offset-free normally distributed errors for the measurements. Although in many cases, probabilistic approaches provide good results and have low computational weight, the uncertainty of the localization estimate is often severely underestimated. The problem with probabilistic approaches is that the error propagation from the sensor readings to the localization results under the Gaussian assumption involves linearization that introduces incorrect approximations in the uncertainty computation. The linearization point significantly influences the uncertainty assessment. Furthermore, the true error distribution usually deviates from the normal distribution, again introducing approximation errors leading to too optimistic uncertainty estimations. Regarding the Stanford-ESA Diagram, underestimating the true error leads to the unsafe operation in the bottom right triangle in Figure 1.1. Hence, an overly optimistic uncertainty estimate means the localization system raises fewer alerts, although the true error may exceed the alert limit. [4, 11, 12]

An alternative model to asses the uncertainty is to use set theory. In this thesis, we will use interval analysis which is a part of set theory and provides convenient tools to work with box-like sets. The basic idea of set-membership approaches is to initially start with a large region where the vehicle is localized and to gradually dismiss infeasible and inconsistent parts in such a way so that we only obtain a set that satisfies the set of applied constraints. The advantage of interval-based approaches is that we do not need to know the distribution of the

sensor errors – but we need to know the upper bounds of the error. If those upper bounds are satisfied, interval-based approaches provide well-defined sets that guarantee to enclose the correct location of the vehicle. However, the main disadvantage of those methods is their pessimism leading to wide sets providing large uncertainties as only worst-case scenarios are modeled by the bounds. Furthermore, no point-valued results are provided by purely set-membership-based methods. The pessimistic uncertainty estimation is assigned to the top left triangle in the Stanford-ESA Diagram in Figure 1.1. The methods highly overestimate the error, although the true error may be significantly smaller. Hence, the system is unavailable for the same alert limit as the user is always warned that it cannot operate since the uncertainty is too high. [4, 11, 12]

In summary, probabilistic approaches often provide good results but tend to underestimate the uncertainty. Interval approaches are promising in the uncertainty assessment but can be very pessimistic, due to which the localization approach may provide inappropriately large uncertainties and not provide point-valued results. Consequently, both approaches have complementary properties. Hence, this observation raises the central research question of this thesis:

- **Is there a symbiotic relationship between set-membership and probabilistic approaches that can be exploited to improve the robot localization estimation and the uncertainty assessment?**

If so, the symbiotic relationship should lead to a mutual improvement of each approach compared to the case they are applied individually. Consequently, two further questions arise that we will focus on in this work:

- **How can set-membership approaches be used to robustify probabilistic approaches?**
- **How can probabilistic approaches be used to reduce the pessimism of set-membership approaches?**

In complement to well-founded mathematical theories such as Possibility theory [13–15] and Dempster-Shafer theory [16, 17], this thesis aims to show different practical ways to combine interval and probabilistic approaches to solve the robot localization problem by providing a unified hybrid approach.

1.3 Solution Approach and Contributions

This work introduces our novel **H**ybrid **P**robabilistic- **a**nd **S**et-Membership-based **Co**arse and **Re**fined (HyPaSCoRe) Localization approach. The HyPaSCoRe Localization combines probabilistic and set-membership approaches into one unified method that localizes a robot in a building map in real-time. The method is composed of three modules: visual odometry, coarse localization, and refined localization. Figure 1.2 shows the method overview.

Our method uses a stereo camera system, Light Detection And Ranging (LiDAR), and Global Navigation Satellite System (GNSS) data. Note that we assume the sensors to be calibrated and synchronized. The HyPaSCoRe Localization focuses on the localization in urban canyons. The main problem of urban regions is that GNSS data can become highly inaccurate as multi-path effects corrupt the GNSS-based location estimates. Consequently, the uncertainty

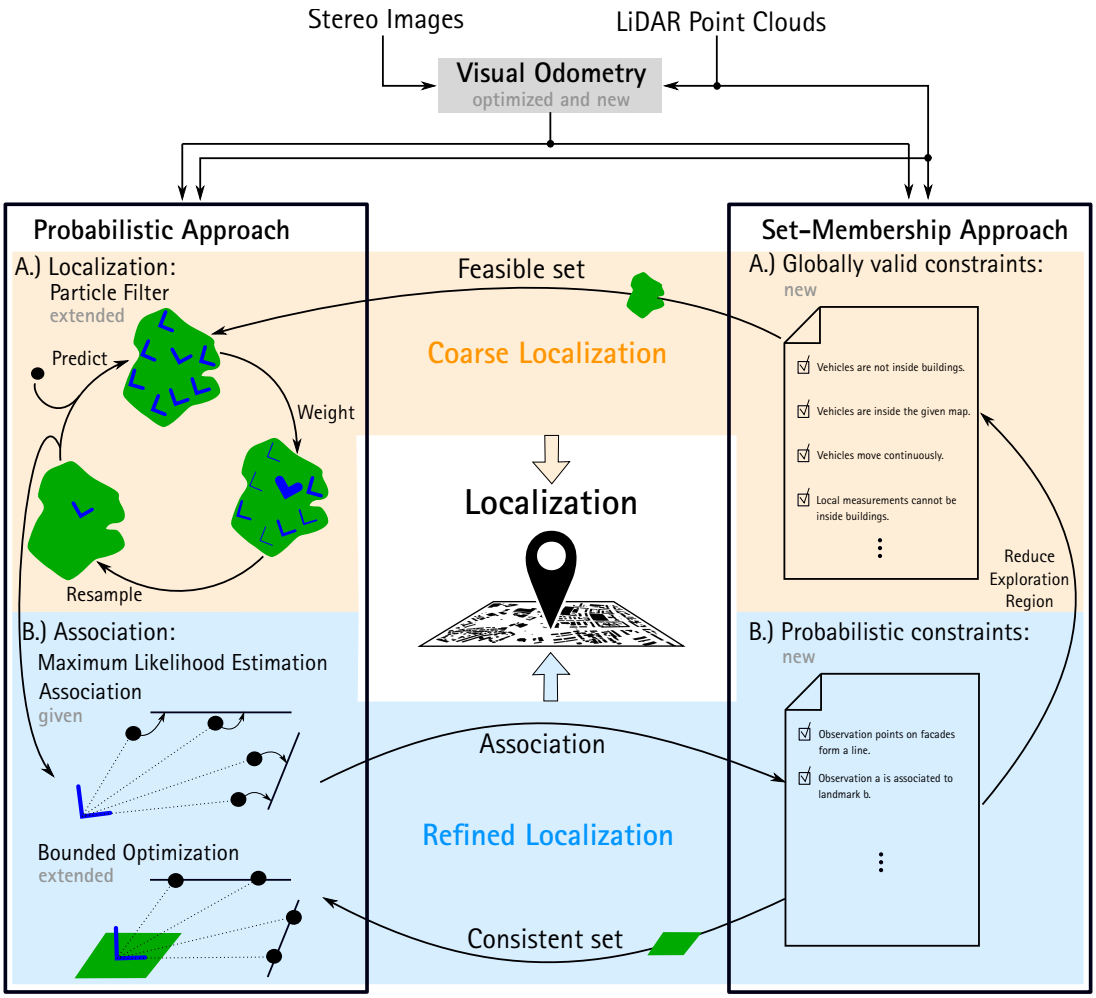


Figure 1.2: Method overview to the HyPaSCoRe Localization pipeline. In gray we specify which parts are given, optimized, extended and new.

of the GNSS measurements is very large, which poses a problem when the localization is initialized. As the vehicle's location is unknown, we must assume that the vehicle is placed anywhere on the map. Although GNSS data provides a first estimate, we can only partially rely on the GNSS data. Our approach can cope with those large uncertainties. It can provide a feasible set of poses that encloses the correct pose and provides a maximum likely pose that best fits the local LiDAR measurements to the building map.

As shown in Figure 1.2, the visual odometry module uses stereo images and LiDAR data to compute the relative motion of the vehicle. The coarse and refined localizations use the relative motion information. The coarse localization module is colored orange and has a set-membership and probabilistic part. The set-membership part uses basic but globally valid constraints to narrow down the feasible set of poses. Such basic constraints include, for instance, that the vehicle cannot be located inside a building. The feasible set is then forwarded to the probabilistic approach. A green region illustrates the feasible set in Figure 1.2. Using a novel bounded Monte Carlo Localization (MCL) with an aggressive resampling procedure, the coarse localization's probabilistic part provides the most likely poses inside the feasible set.

The bottom blue part of Figure 1.2 illustrates the refined localization module that further refines the coarse localization result. Since the coarse localization approach does not consider

the direct associations of the locally captured data to the building map, the feasible set is comparatively pessimistic. The refined localization aims at reducing the pessimism of the uncertainty estimate. Therefore, the probabilistic part of the refined localization chooses the best fitting particle of the bounded MCL of the coarse localization. Based on the point-valued pose estimate, the local LiDAR data is associated with the building facades on the map. We illustrate this procedure with black dots for the LiDAR points and black lines for the facades. The association is forwarded to the set-membership part of the refined localization as we can draw local association constraints. Those association constraints considering the building map uncertainty and the LiDAR measurement uncertainty are used to determine the consistent set. The bottom arrow from the set-membership approach to the probabilistic approach indicates that the consistent set provides bounds for a novel modified bounded optimization approach that determines the most likely pose within the smaller consistent set. The consistent set determined by the set-membership approach prevents the probabilistic approach from significantly diverging by limiting the solution space.

Note the most right arrow in Figure 1.2 that connects the consistent set of the refined localization with the pessimistic coarse localization: In the case of high reliability of the consistent set, we contract the feasible set to the consistent set to reduce the pessimism of the uncertainty estimate. This contraction automatically reduces the exploration region for the bounded MCL in the probabilistic part of the coarse localization and thereby conditions and robustifies the bounded MCL. From the HyPaSCoRe Localization, we obtain a feasible set as the uncertainty estimation and the most likely pose as the best point-valued localization result. The HyPaSCoRe Localization is real-time capable and is operable in different environments as long as enough buildings are visible.
The core contributions of this work are:

- Investigation of the symbiotic relationship between set-membership-based and probabilistic localization methods to overcome the shortcomings of the individual approaches.
- Development of the novel visual odometry that combines windowed bundle adjustment with an interval-based odometry computation. The probabilistic approach provides the probabilistic constraints, while the set-membership approach implements the error propagation.
- Design of the novel coarse localization method that can cope with large GNSS uncertainties combining a set-membership method with a probabilistic approach in a hybrid fashion.
- Development of the novel refined localization that improves the localization estimates reducing the pessimism by introducing probabilistic maximum likely association and interval bounded optimization.
- Design of the novel HyPaSCoRe Localization pipeline that localizes a vehicle in publicly available building maps using the coarse localization and refined localization.
- Evaluation of all modules, including visual odometry, coarse and refined localization in an ablation study. Furthermore, we extensively evaluate the HyPaSCoRe Localization with real author-collected and publicly available benchmark datasets.
- A software package that implements the method is made publicly available here: `https://github.com/AaronEhambram/hypascore_localization`.

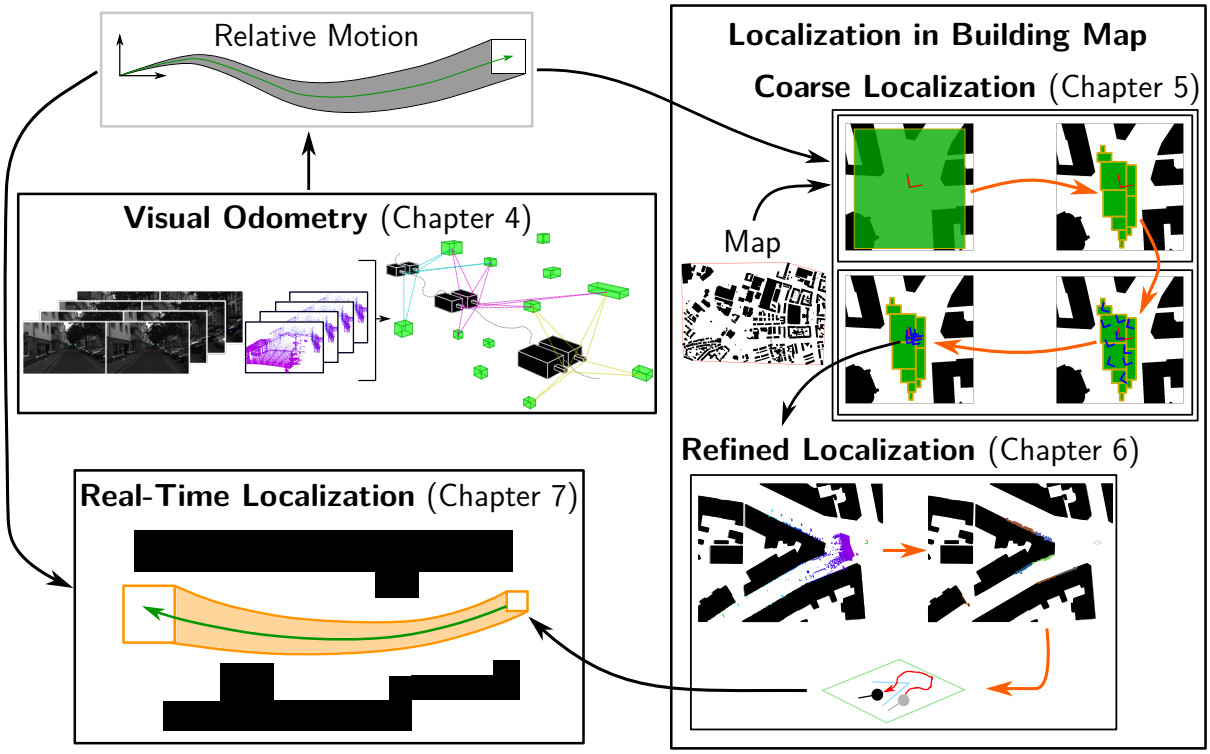


Figure 1.3: Overview of the thesis structure.

1.4 Structure of the Thesis

Figure 1.3 shows the graphical overview of this thesis containing the different modules developed in this work. Throughout this document, we will use this figure to put the different methodological chapters into the overall HyPaSCoRe Localization context. While **Chapter 2** introduces the required basics for this work, **Chapter 3** summarizes the State of the Art for robot localization.

In **Chapter 4**, we introduce the first module of the HyPaSCoRe Localization system. The corresponding block in Figure 1.3 is presented on the left. As indicated, it provides the relative motion information. **Chapter 5** describes the coarse localization. As illustrated in Figure 1.3 it uses the relative motion information to provide a coarse localization estimate considering large uncertainties. It is the first part of the localization block. The refined localization in the bottom part of the localization block in Figure 1.3 refines the coarse localization results and is explained in **Chapter 6**. The full system architecture that combines the relative motion estimates with the real-time localization in the HyPaSCoRe Localization is detailed in **Chapter 7** and illustrated by the bottom left block in Figure 1.3.

The experimental evaluation based on different benchmark datasets and the comparison of our approach and selected State of the Art approaches is provided in **Chapter 8**. In **Chapter 9**, we discuss the results obtained in this work. Furthermore, we identify open research questions and suggest future work. Finally, **Chapter 10** summarizes and concludes this thesis.

2 Basics

In robotics, many problems can be generalized to state estimation problems. For instance, the robot localization problem is a typical state estimation problem, where the state we seek to estimate is the vehicle's pose. The core difficulty is that we typically do not have an accurate sensor to measure the pose directly. Instead, the robot is equipped with sensors that perceive the environment locally so that by associating the local measurements to the map, we aim to deduce the robot's state. However, local measurements are not perfect. Depending on the sensors the robot is equipped with, we have to deal with measurement errors which we try to model with uncertainty representations. How do the uncertainties of the local measurements affect the state that we deduce from them? We must propagate the local measurement uncertainties to the state to answer this question by applying the state equations.

The details of the uncertainty propagation depend on the uncertainty model. In this work, we present two fundamentally different approaches – namely, the classical probabilistic and the interval-based model. In Section 2.1 we introduce the fundamental basics of probability theory and in Section 2.2 interval analysis. Furthermore, this chapter will also present the working principle of the used sensors and their sensor models in Section 2.3. We conclude this chapter with the structure of building maps in Section 2.4 that are used in this work.

2.1 Probability Theory

The key concept in probabilistic robotics is to represent uncertainties using probability theory. While earlier works in the field typically tried to come up with single best guesses, probabilistic approaches focus on constructing the probability distribution over a set of solutions [18]. In the robotics community, probabilistic approaches have found broad acceptance since they represent the concept of risk in a mathematically sound way, assuming that certain assumptions are met. Although probabilistic robotics, in its basic concept, is indeed a mathematically elegant way to handle uncertainties, concrete implementations are fraught with significant shortages and problems. This thesis aims to partially overcome those problems by combining probabilistic approaches with interval analysis.

We briefly introduce some basic probability theory to understand the problems that probabilistic approaches may run into and how we can use interval analysis to counteract such problems. Afterward, concrete and well-known implementations of probabilistic approaches are evaluated, pointing out the weaknesses. The following notions and definitions are taken from Sebastian Thrun's book [18].

2.1.1 Basic Notions and Concepts

The central idea of probabilistic robotics is to portray quantities such as sensor measurements, controls, the states of the robot (e.g. the pose), and the environment as random variables. The strategy to harness random variables to propagate information from one set of random variables to another is based on fundamental probability theory. Consider a random variable X.

Definition 2.1.1

A random variable X is a measurable function $X : \Omega \to E$ from a sample space Ω as a set of possible outcomes to a measurable space (Borel space) E. The probability that X takes on a value in a measurable set $S \subseteq E$ is written as

$$P(X \in S) = P(\{w \in \Omega | X(w) \in S\}). \tag{2.1}$$

Example 2.1.1. For example, rolling a dice can be modeled as a random process, where a random variable defines the state of the dice. The state of the dice can be represented by the side that is upward facing after rolling. In this case, the random variable can take values between 1 and 6.

When the image of X countable, the random variable is called discrete random variable as it is the case in Example 2.1.1. Its distribution is a discrete probability distribution described by a probability mass function. However, if the image of X is uncountably infinite, then X is called a continuous random variable. Its distribution can be described by a probability density function (PDF).

First, let us consider the discrete case. We denote x as a specific value the random variable X might assume. The probability that X has value x is denoted by $P(X = x)$. Considering Example 2.1.1, the probability that X takes one value between 1 and 6 is equally distributed in the case of a fair dice, and the probability is $P(X = 1) = ... = P(X = 6) = \frac{1}{6}$. The random variable has to take a value among the possible events. This means, for Example 2.1.1, the probability that the dice will take one of the states 1 to 6 is 1. Mathematically speaking, the sum over all probabilities among all values that a random variable can take always sums to 1:

$$\sum_x P(X = x) = 1. \tag{2.2}$$

Note that probabilities are always non-negative.

For the sake of simplicity, we want to rewrite $P(X = x)$ by $P(x)$. While Example 2.1.1 introduces the model of a discrete event space with an equal probability distribution among all events, continuous spaces are characterized by random variables that can take on a continuum of values. The mapping function

$$p(x) \ : \ \mathbb{R} \to [0, 1]. \tag{2.3}$$

defines the probability density among the continuous spaces. In the scalar case it can, for instance, be $x \in \mathbb{R}$. If the event space is continuous, we need to adapt (2.2) by taking the integral into account

$$\int_x p(x)dx = 1. \tag{2.4}$$

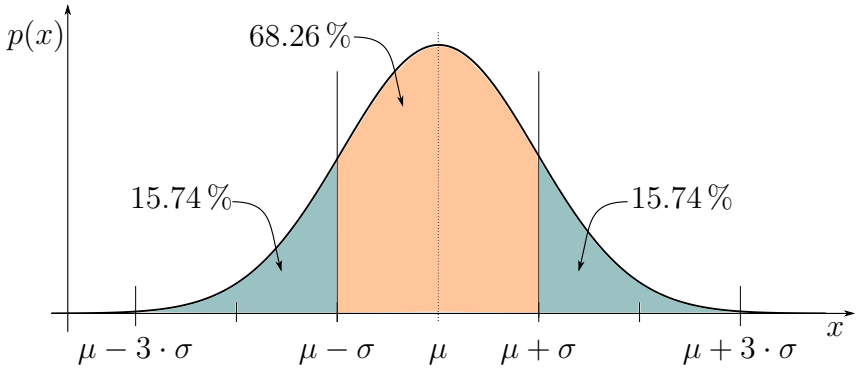


Figure 2.1: One-dimensional normal distribution. The size of the confidence intervals of defined confidence probabilities depends on the standard deviation σ, while the mean μ determines the probability peak. The distribution is symmetrical to the mean.

A PDF can generally have an arbitrary shape as long as the properties mentioned above are fulfilled. However, dealing with arbitrary PDFs is often burdened with high computational efforts when it comes to inference – that means if we want to deduce the probability of the events in question. Fortunately, there are also PDFs that have convenient mathematical properties that make efficient inference possible. For example, the normal distribution has such convenient properties. Additionally, as in robotic applications the measurement error is often composed of many independent additive errors, the Central Limit Theorem [10, 19] legitimates the employment of the normal distribution and has proven to be a good choice for uncertainty modeling in robotics.

The normal distribution is characterized by the first two moments – the mean μ and the variance σ. The Gaussian function defines the normal distribution

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2.5}$$

Figure 2.1 visualizes the one-dimensional normal distribution. Note that this PDF is fully determined by the two parameters μ and σ. One of the nice properties the normal distribution has is that so-called confidence intervals can easily be constructed. Confidence intervals determine an interval of values the random variable X might take for a defined confidence probability. As illustrated in Figure 2.1, the width of such confidence intervals is determined by the standard deviation σ. Hence, the probability that X takes a value in the interval $[\mu - 1 \cdot \sigma, \mu + 1 \cdot \sigma]$ has a confidence probability of $\int_{\mu-1\cdot\sigma}^{\mu+1\cdot\sigma} p(x)dx = 68.26\,\%$, while a wider interval like for instance $[\mu - 3 \cdot \sigma, \mu + 3 \cdot \sigma]$ has a confidence probability of $\int_{\mu-3\cdot\sigma}^{\mu+3\cdot\sigma} p(x)dx = 99.73\,\%$. As we will see later in this work, confidence intervals will be the key to combining interval-based set-membership and probabilistic approaches.

(2.5) describes the normal distribution in the one-dimensional case. However, often $\mathbf{x}$ will be a multi-dimensional vector. Normal distributions over multi-dimensional vectors are called multivariate. The multivariate normal distribution is described by the density function

$$p(\mathbf{x}) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \cdot e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{2.6}$$

Note that (2.6) is a strict generalization of (2.5). While $\boldsymbol{\mu}$ is the mean vector, Σ is the so-called covariance matrix directly linked to σ^2 in the one-dimensional case.

Let us turn our attention to discrete probability distributions again. The joint distribution of two random variables is given by $P(x\,,\,z)$, which describes the probability, that the random variable X takes the value x and Z the value z. Conditional probabilities are described by $P(x\,|\,z)$, which provides the probability of the occurrence of x for X given we know that Z's value is z. Joint and conditional probabilities are linked with each other by

$$P(x\,|\,z) = \frac{P(x\,,\,z)}{P(z)}. \tag{2.7}$$

Now we can introduce *Bayes rule*, which plays a predominant role in probabilistic inference in robotics. It relates a conditional probability of the type $P(x\,|\,z)$ to its "inverse" $P(z\,|\,x)$. The rule

$$P(x\,|\,z) = \frac{P(z\,|\,x)\cdot P(x)}{P(z)} \tag{2.8}$$

requires $P(z) > 0$. We name $P(x)$ the prior probability distribution, x represents the quantity we want to infer (e.g. pose of the robot), and z represents the sensor measurement. The distribution $P(x)$ describes our knowledge before incorporating the sensor measurements z. The probability $P(x\,|\,z)$ is the posterior probability distribution that interests us the most since it describes the probability of x given the specific sensor measurements. The Bayes rule provides a convenient way to compute this probability using the inverse conditional probability $P(z\,|\,x)$ and the prior probability distribution $P(x)$. The inverse conditional probability $P(z\,|\,x)$ describes the probability of sensor measurement z given the robot state x. Since the probability describes how a specific state x of the robot causes sensor measurements y, this probability is determined by the sensor model known a priori and the prior knowledge on x.

In our context, the state vector x can contain static landmarks in the environment and/or the poses of the robot at different times. We denote a state at time t by x_t. Furthermore, we define z_t as the sensor measurements at time t. Usually, in robotics, we also have access to the control commands u_t at time t, providing valuable information that can be used to infer the robot's state x_t. The goal in robotics is to estimate the current state of the robot x_t given all previous states $x_{1:t-1}$, all previous sensors measurements $z_{1:t-1}$ and control commands $u_{1:t}$. As a consequence, we want to determine the PDF of $p(x_t\,|\,x_{1:t-1}, z_{1:t-1}, u_{1:t})$.

Historically, in localization and mapping, two paradigms mainly influenced the field. Filter approaches mainly defined the first paradigm. The core assumption here is that the modeled states are complete: That means that knowledge of past states, measurements, or controls do not provide additional information to help us predict the future more accurately than the last state. We only need to know the previous state and can forget everything that happened before. We call such temporal processes Markov chains. The Bayes Filter is the most general algorithm that calculates the probability distribution of the current state of the robot based on the previous state, the measurements, and control data. The Gaussian implementation of the Bayes Filter is the Extended Kalman Filter (EKF). Since the general Bayes Filter and the EKF are not directly relevant to our work, we provide a brief summary in the appendix in

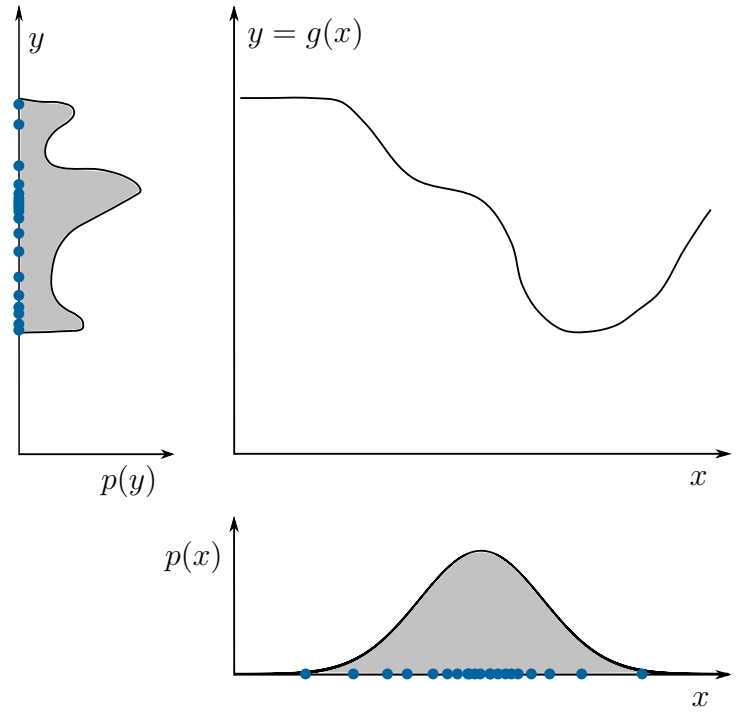


Figure 2.2: Particle representation of distributions based on [18]. The PDF of the input variable x is Gaussian distributed. Instead of representing the distribution in its parametric form, the particle filter approximates the distribution by particles visualized by blue dots. The particle filter performs the propagation through the non-linear function $g(x)$ by evaluating each particle individually. This makes the particle filter well applicable to arbitrary distributions and non-linear functions. The distribution of the particles of the output random variable well approximates the PDF, as illustrated on the left.

Section A.1 and Section A.2. The particle filter introduced in Section 2.1.2 is a non-parametric form of the Bayes Filter and will play an essential role in the HyPaSCoRe Localization pipeline.

The hallmark of the second paradigm is optimization – sometimes also called smoothing. The approaches perform batch optimization and typically rely on least-square error minimization [20]. In contrast to filtering approaches, optimization methods do not assume states to be complete. As a result, older states, measurements, and control data are also involved in the computation of the current robot state [21]. A non-linear least squares approach is used to determine the posterior probability distribution, which is introduced in Section 2.1.3.

2.1.2 Particle Filter

The particle filter is a non-parametric implementation of the Bayes Filter. Non-parametric means that the underlying but unknown PDFs are not represented by a set of parameters. Instead, the particle filter represents a distribution by a set of samples drawn from the distribution. Although this representation is approximate, the non-parametric description makes the particle filter applicable to a broader family of PDFs. Figure 2.2 illustrates this for the same example shown in Figure A.1 for the EKF in the appendix.

Samples of a distribution are called particles and are denoted by $\mathcal{X}_t = \{x_t^1, ..., x_t^n\}$ for $n \in \mathbb{N}$ particles. Particles can be seen as a actual hypothesis of the state at time t. The intuition behind particle filters is to sustain the belief bel(x_t) by a set of particles $\mathcal{X}_t$. As illustrated in Figure 3.2, the denser particles populate a subregion, the higher the probability that the true state lies in this subregion. Just as all Bayes Filter algorithms, the particle filter constructs the belief of the current state bel(x_t) recursively from the belief of the previous state bel(x_{t-1}). Therefore, the particle filter also performs a prediction and correction step. A simple form of the particle filter algorithm is shown in Algorithm 1.

Algorithm 1: Particle Filter

Data: $\mathcal{X}_{t-1}$, u_t, z_t
Result: $\mathcal{X}_t$
1 $\overline{\mathcal{X}}_t = \mathcal{X}_t = \varnothing$;
`// Prediction step`
2 **for** *all* x_{t-1}^i *in* $\mathcal{X}_{t-1}$ **do**
3 sample $\overline{x}_t^i$ from $p(x_t \mid u_t, x_{t-1}^i)$;
4 $w_t^i = p(z_t \mid \overline{x}_t^i)$;
5 add $\{\overline{x}_t^i, w_t^i\}$ to $\overline{\mathcal{X}}_t$;
6 **end**
`// Correction/Resampling step`
7 **for** $m = 1$ *to* n **do**
8 draw i with the probability w_t^i;
9 add $\overline{x}_t^i$ to $\mathcal{X}_t$;
10 **end**

Typically starting with a uniform distribution of the particles as a first approximation, the true picture of the PDF emerges as the result of an iterated prediction and resampling process, which takes into account the weights of particles induced by the fitness of local measurements to the particle location in the map.

The prediction step generates new particles $\overline{x}_t^i$ from the old particles x_{t-1}^i by randomly updating the old particles based on the control u_t. For example, in the robot localization problem, the prediction step is typically implemented as the particle update step that applies the odometry measurements to the old particles. In line 4, the predicted particle is evaluated, considering the sensor measurement z_t. The evaluation provides an importance weight for the predicted particle $\overline{x}_t^i$. The higher the weight, the better the sensor measurements comply with the predicted state. Hence, the particle is evaluated as very likely if the weight is high.

In the correction step, the "trick" of the particle filter happens. Note that we only use the term *correction* to maintain the link to the general structure of the Bayes Filter. However, in the case of the particle filter, the correction step is a resampling step where we randomly resample from the predicted set of particles $\overline{\mathcal{X}}_t$ another set $\mathcal{X}_t$ of n particles. The resampling procedure is implemented as an importance sampling where the importance of a particle is determined by its weight, which we computed in the prediction step. That means if a particle $\overline{x}_t^i$ is evaluated as likely, the particle's weight will be high. Accordingly, the probability that this particle is resampled from $\overline{\mathcal{X}}_t$ is higher. Consequently, after the filtering step, only those particles will be considered for the next iteration step that are evaluated as very likely. By incorporating the importance weights into the resampling process, the distribution of the particles changes.

While before the resampling step, the particles are distributed according to $\overline{\text{bel}}(x_t)$, after the resampling, they are distributed approximately according to the posterior $\text{bel}(x_t)$.

In contrast to the EKF, the particle filter can deal with arbitrary distributions and non-linear functions. However, the accuracy of the calculated posterior probability critically depends on the number of particles. The more particles are used for the approximation, the more accurate the approximation will be. Nonetheless, more particles directly imply higher computational effort since each particle needs to be evaluated individually. As a result, a trade-off between accuracy and computation effort is necessary. Moreover, due to the random resampling procedure, an unfortunate sampling sequence can lead to particle depletion. This problem happens if particles cluster in regions with low probability, and the resampling can lead particles to vanish or diverge from the correct solution. To overcome the problems, many different approaches were suggested in the literature, and the interested reader might consult [18].

2.1.3 Optimization

While filtering approaches model the localization and mapping problem as an online state estimation, where the system consists of only the *current* robot pose and the map, optimization approaches estimate the entire trajectory of the robot from the complete set of measurements. Hence, optimization approaches rely on least-square error minimization techniques instead of incrementally updating the state as filters do.

A graph-based formulation of the problem has proven to be intuitive for localization and mapping tasks. Especially in the context of SLAM problems, graph-based approaches for optimization have become the State of the Art. While SLAM is not our focus here, the graph-based formulation will play an essential role in this work. Therefore, this section provides an introduction to graph-based SLAM. Graph-based SLAM solutions solve the full SLAM problem consisting of estimating the posterior probability of the robot's whole trajectory $x_{1:t}$ and the map m of the environment given all the measurements and the initial pose x_0: $p(x_{1:t}, m \,|\, z_{1:t}, u_{1:t}, x_0)$. This is the core difference to filtering approaches that only consider the last state, ignoring the trajectory before.

In graph-based SLAM, poses of the robot and the position of landmarks are modeled by nodes in a graph. Spatial constraints between poses and landmarks resulting from odometry measurements u_t or landmark observations z_t are represented by edges between the respective nodes. Figure 2.3 shows an exemplary SLAM-graph. Solving a graph-based SLAM problem can be decoupled into two subtasks. The first subtask consists of the constructing the graph from the raw measurements. This step is usually called the front-end. The second subtask, given the edges of the graph, the most likely configuration of the poses and landmarks is determined. This step is called back-end.

The front-end typically performs the data association and relates the raw measurements with edges in the graph that constrain nodes. Ideally, this step involves the modeling of a full-fledged random process. Yet, this strategy typically encounters a stumbling block: The modeling of the data association gets mired in a combinatorial explosion. That is why usually, the data association takes the shortcut of the maximum likely association, suppressing lower probability associations. The goal in the back-end of the graph-based approach is to approximate the posterior over the robot trajectory and the map. Under the assumption that the observations

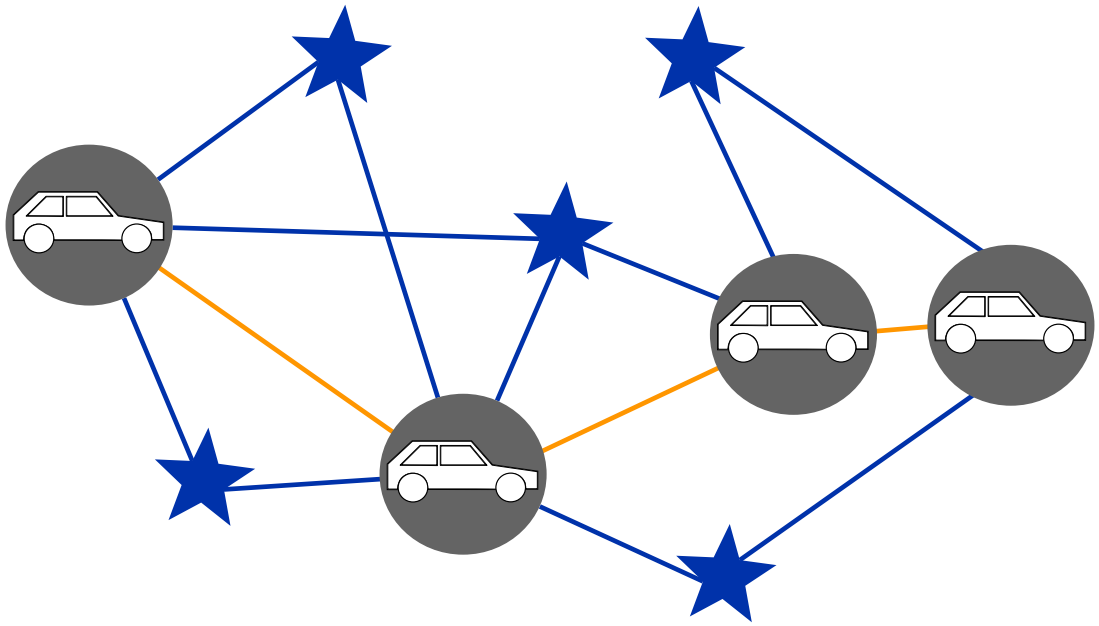


Figure 2.3: Exemplary SLAM-graph. Poses and landmarks represent the parameter blocks that need to be determined. Those parameter blocks are represented in the graph by nodes. Grey nodes with the car symbol visualize pose nodes, and blue star-shaped nodes represent the landmarks. The edges between the nodes are measurements that represent constraints between the connected nodes. The blue edges illustrate observation constraints between pose and landmark nodes. The orange edges illustrated odometry constraints between consecutive pose nodes.

are affected by Gaussian noise only and the data association in the front-end is correct, the mean of the Gaussian that describes the posterior that we are interested in can be determined by computing the configuration of the nodes that maximizes the likelihood of the observations. Since we assume Gaussian observation uncertainty, the Maximum Likelihood Estimation (MLE) of the configuration can be cast into a least-squares minimization problem that sophisticated numerical solvers can cope with. While the front-end depends on the type of sensors we are using, the back-end performs the probabilistic estimation process based on the abstract graph representation. We will examine the back-end in the following to understand the strengths and drawbacks of graph-based optimization approaches.

Let $\mathbf{x} = \left(\mathbf{x}_1^T, ..., \mathbf{x}_n^T\right)^T$ be a vector of parameters, where each $\mathbf{x}_i$ represents a generic parameter vector. The parameter vector may represent, for instance, the robot's pose at a certain time step or a landmark that we inserted into the graph in the front-end. For example, if we consider a 6DOF pose, six parameters need to be considered, and in the case of a point landmark, three parameters are considered by $\mathbf{x}_i$. Let $\mathbf{z}_{ij}$ and Ω_{ij}^{-1} represent respectively the mean and the covariance of a measurement that relates $\mathbf{x}_i$ and $\mathbf{x}_j$. While we interpret the raw sensor measurements as mean values, the sensor manufacturer provides the measurement uncertainty described by covariances. Typically, $\mathbf{x}_i$ is a pose and $\mathbf{x}_j$ a landmark. In that case, $\mathbf{z}_{ij}$ describes how the landmark $\mathbf{x}_j$ was seen from the pose $\mathbf{x}_i$ and Ω_{ij}^{-1} describes the uncertainty of the observation provided by the sensors. However, both nodes may also describe poses. Then, $\mathbf{z}_{ij}$ would be the odometry measurement. As a consequence, $\mathbf{z}_{ij}$ can be seen more

generally as a constraint between $\mathbf{x}_i$ and $\mathbf{x}_j$, and the accuracy of the constraint is defined by Ω_{ij}.

The error function $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ measures how well the parameters $\mathbf{x}_i$ and $\mathbf{x}_j$ satisfy the constraint $\mathbf{z}_{ij}$. Therefore, the error function computes the difference between the expected observation $\hat{\mathbf{z}}_{ij}$ for the given $\mathbf{x}_i$ and $\mathbf{x}_j$ and the real observation $\mathbf{z}_{ij}$. Note that the expected observation $\hat{\mathbf{z}}_{ij}$ comes from the sensor model that we defined for the EKF and the particle filter as $h(x_t)$ (cf. Subsection A.2 and 2.1.2). As a result, the error is defined by

$$\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j). \tag{2.9}$$

Since the $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ relates to a measurement $\mathbf{z}_{ij}$ with the covariance matrix Ω_{ij}^{-1}, the contribution of this error to the objective function that we seek to minimize is

$$\mathbf{F}_{ij} = \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^T \Omega_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}). \tag{2.10}$$

The quadratic formulation of the error in (2.10) results from the goal to determine the distribution of $p(x_{1:t}, m \mid z_{1:t}, u_{1:t}, x_0)$. A satisfactory way to achieve that is the MLE. As shown in [10], if all related random variables of a problem are Gaussian distributed, the MLE problem becomes a least squares problem. Consequently, the quadratic formulation of the error in (2.10) is only valid if we assume Gaussian uncertainties for the measurements. That means the core assumption of the optimization approach is the Gaussian distribution of all related variables.

The full objective function is the sum of all quadratic errors that are considered in the graph encoded as observation constraints in the edges

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{F}_{ij} \tag{2.11}$$

The goal in the back-end of the graph-based optimization approach is to find a set of parameters $\mathbf{x}^*$ that minimizes the objective function so that

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \mathbf{F}(\mathbf{x}). \tag{2.12}$$

So far, we just considered how the graph optimization problem can be formulated. The following will examine how this least-squares problem in (2.12) can be solved. If a good initial guess $\check{\mathbf{x}}$ is known that is comparatively close to the optimal solution $\mathbf{x}^*$, the numerical solution of (2.12) can be obtained by using popular Gauss-Newton or Levenberg Marquardt (LM) algorithms [22]. The idea of such numerical solvers is to iteratively minimize the error by applying gradient descent. Therefore, the error function is approximated by its first order Taylor expansion – as introduced in the EKF – around the current initial guess $\check{\mathbf{x}}$

$$\mathbf{e}(\check{\mathbf{x}}_i + \Delta\mathbf{x}_i, \check{\mathbf{x}}_j + \Delta\mathbf{x}_j, \mathbf{z}_{ij}) =: \mathbf{e}_{ij}(\check{\mathbf{x}} + \Delta\mathbf{x}) \tag{2.13}$$

$$\approx \mathbf{e}_{ij}(\check{\mathbf{x}}) + \mathbf{J}_{ij}\Delta\mathbf{x}. \tag{2.14}$$

The Jacobian $\mathbf{J}_{ij}$ of $\mathbf{e}_{ij}(\mathbf{x})$ is computed in $\check{\mathbf{x}}$. Substituting (2.14) in the error term $\mathbf{F}_{ij}$ in (2.10) leads to

$$\begin{aligned}
\mathbf{F}_{ij}(\check{\mathbf{x}} + \Delta\mathbf{x}) &= \mathbf{e}_{ij}(\check{\mathbf{x}} + \Delta\mathbf{x})^T \Omega_{ij} \mathbf{e}_{ij}(\check{\mathbf{x}} + \Delta\mathbf{x}) && (2.15)\\
&\approx (\mathbf{e}_{ij} + \mathbf{J}_{ij}\Delta\mathbf{x})^T \Omega_{ij} (\mathbf{e}_{ij} + \mathbf{J}_{ij}\Delta\mathbf{x}) && (2.16)\\
&= \underbrace{\mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij}}_{c_{ij}} + 2\underbrace{\mathbf{e}_{ij}^T \Omega_{ij} \mathbf{J}_{ij}}_{\mathbf{b_{ij}}^T} \Delta\mathbf{x} + \Delta\mathbf{x}^T \underbrace{\mathbf{J}_{ij}^T \Omega_{ij} \mathbf{J}_{ij}}_{\mathbf{H}_{ij}} \Delta\mathbf{x} && (2.17)\\
&= c_{ij} + 2\mathbf{b}_{ij}^T \Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{H}_{ij} \Delta\mathbf{x}. && (2.18)
\end{aligned}$$

For sake of simplicity we abbreviate $\mathbf{e}_{ij}(\check{\mathbf{x}})$ by $\mathbf{e}_{ij}$. If we substitute the local approximation of $\mathbf{F}_{ij}$ in the full objective function (2.11) that considers all edges in the graph, we obtain

$$\begin{aligned}
\mathbf{F}(\check{\mathbf{x}} + \Delta\mathbf{x}) &= \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{F}_{ij}(\check{\mathbf{x}} + \Delta\mathbf{x}) && (2.19)\\
&\approx \sum_{\langle i,j \rangle \in \mathcal{C}} c_{ij} + 2\mathbf{b}_{ij}^T \Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{H}_{ij} \Delta\mathbf{x} && (2.20)\\
&= c + 2\mathbf{b}^T \Delta\mathbf{x} + \Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x}. && (2.21)
\end{aligned}$$

By setting $c = \sum c_{ij}$, $\mathbf{b} = \sum \mathbf{b}_{ij}$ and $\mathbf{H} = \sum \mathbf{H}_{ij}$, we obtain the quadratic form of the linearized error as presented in (2.21). Since we aim to minimize the error, we seek to find the increment $\Delta\mathbf{x}$ that minimizes (2.21). The optimal value of $\Delta\mathbf{x}$ is determined by solving linear system

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}. \tag{2.22}$$

The obtained solution for $\Delta\mathbf{x}$ is just an increment that improves the initial guess of the values of the involved variables by reducing the error term. The new guess is $\check{\mathbf{x}} + \Delta\mathbf{x}$. However, the new guess is not necessarily an acceptable solution. As a consequence, the whole procedure of linearizing at $\check{\mathbf{x}} + \Delta\mathbf{x}$ (cf. (2.21)) and solving the linear system (2.22) needs to be repeated until a defined termination criterion is met (Gauss-Newton algorithm). That means the optimization approach computes incremental update steps and linearizes point-wise for each iteration. The best solution for the problem defined in (2.12) is obtained by adding all increments $\Delta\mathbf{x}^*$ to the initial guess

$$\mathbf{x}^* = \check{\mathbf{x}} + \Delta\mathbf{x}^*. \tag{2.23}$$

A more elegant way to solve the linear system in (2.22) is to solve the damped version

$$(\mathbf{H} - \lambda\mathbf{I})\Delta\mathbf{x} = -\mathbf{b}. \tag{2.24}$$

This solver extends the Gauss-Newton method and is called the Levenberg-Marquardt (LM) method. The factor λ introduces further damping and backup actions to the basic Gauss-Newton to control the convergence. For more information on the LM method, consult [23, 24].

The remaining uncertainty of the obtained solution is represented by $\mathbf{H}$ – the information matrix of the system. As defined above, $\mathbf{H}$ is obtained by summing up the matrices $\mathbf{J}_{ij}^T \Omega_{ij} \mathbf{J}_{ij}$ for each constraint. Hence, the structure of $\mathbf{H}$ depends on the Jacobian of the error function.

Since the error function of the constraint only takes the values of two nodes into account, the Jacobian for all parameters $\mathbf{x}$ has the form

$$\mathbf{J}_{ij} = \begin{pmatrix} \dots & \mathbf{0} & \mathbf{A}_{ij} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{B}_{ij} & \mathbf{0} & \dots \end{pmatrix}. \tag{2.25}$$

The terms $\mathbf{A}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_i}|_{\mathbf{x}=\check{\mathbf{x}}}$ and $\mathbf{B}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_j}|_{\mathbf{x}=\check{\mathbf{x}}}$ are the partial derivatives of the error function concerning $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively. All other partial derivatives concerning the other parameters are $\mathbf{0}$, as they play no part in the error term derived from the edge between the graph's i-th and j-th node.

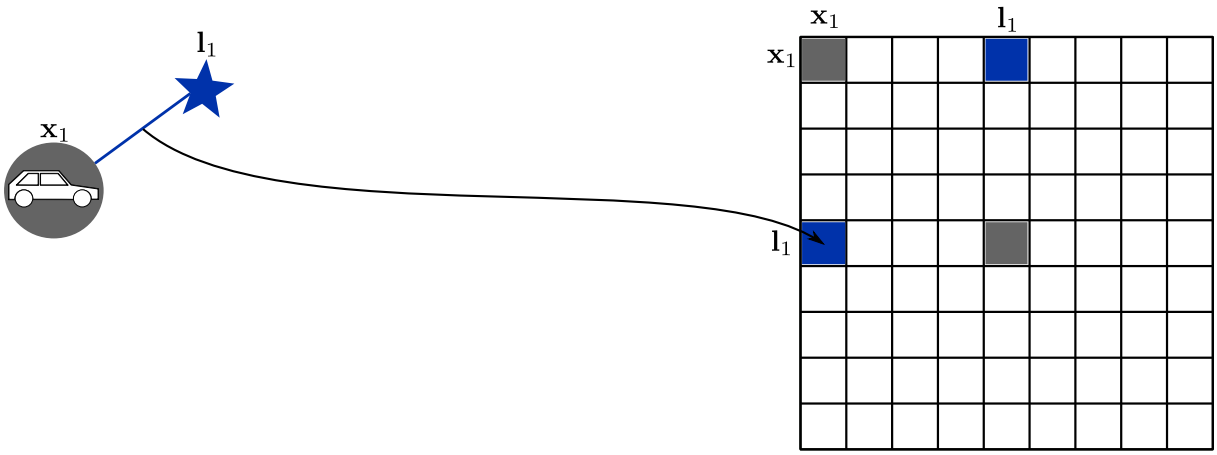
Consequently, the $\mathbf{H}_{ij}$ has a block structure with non-zero entries for the parameters that represent the i-th and j-th node

$$\mathbf{H}_{ij} = \begin{pmatrix} \ddots & & & \\ & \mathbf{A}_{ij}^T \Omega_{ij} \mathbf{A}_{ij} & \dots & \mathbf{A}_{ij}^T \Omega_{ij} \mathbf{B}_{ij} \\ & \vdots & & \vdots \\ & \mathbf{B}_{ij}^T \Omega_{ij} \mathbf{A}_{ij} & \dots & \mathbf{B}_{ij}^T \Omega_{ij} \mathbf{B}_{ij} \\ & & & \ddots \end{pmatrix}. \tag{2.26}$$

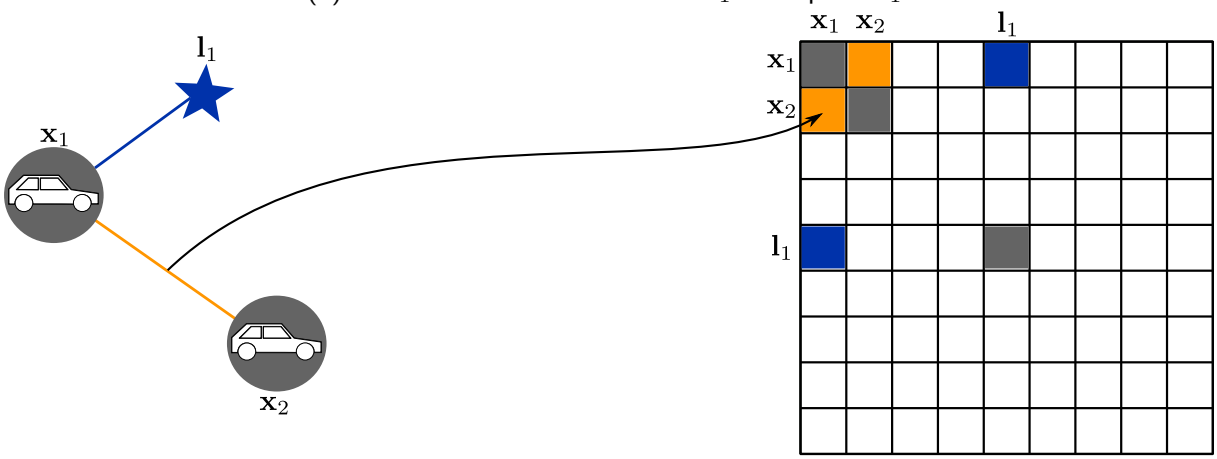
$\mathbf{H}_{ij}$ has a very sparse structure since most entries are zero. As a consequence, the system information $\mathbf{H}$, which is the sum over all $\mathbf{H}_{ij}$, is also a sparse block matrix and is the adjacency matrix of the graph. Figure 2.4 illustrates the structure of the matrix $\mathbf{H}$.

The inverse of the system information matrix $\mathbf{H}$ is the covariance matrix that describes the uncertainty of the estimated parameters. Note that the inversion of the sparse adjacency matrix may result in a densely populated matrix. That means the inversion may become very costly in computation for a large matrix $\mathbf{H}$. While the system covariance matrix considers the correlation between all parameters that may not be directly linked in the graph, those correlations are often negligible. Consequently, the covariance of dedicated parameter blocks can directly be extracted from the system covariance matrix by suppressing the correlation to other parameter blocks (marginalization).

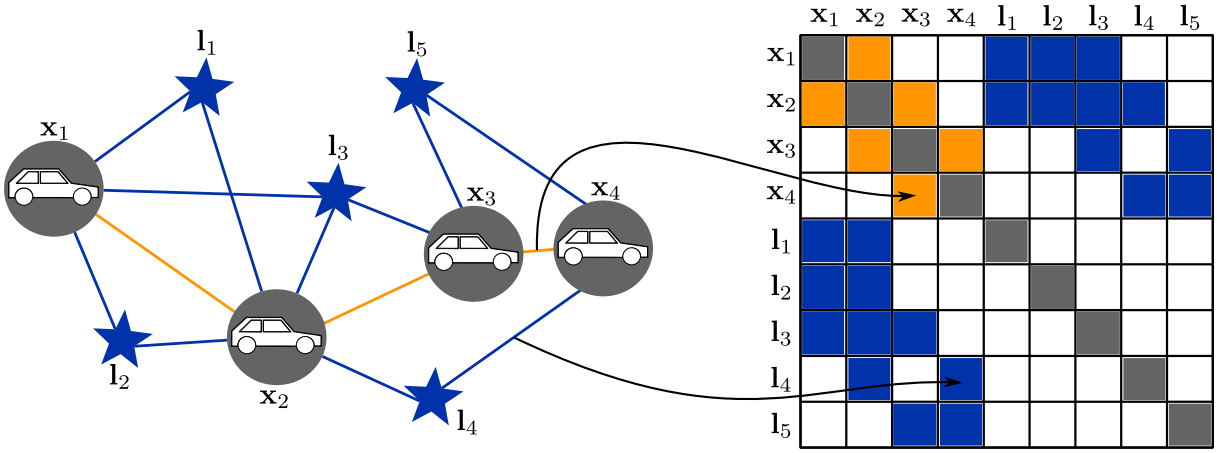
Optimization-based approaches also propagate the uncertainty of the measurements to the estimated state through the Jacobian as the EKF does. Although the optimization performs a linear approximation at each iteration step and can deal with non-linearities better than the EKF, which only performs the linearization for the predicted state, the uncertainty propagation is still potentially fraught with poor approximation. The optimization approach assumes – similar to the EKF – the input and out uncertainties to be Gaussian distributed. However, this assumption is only valid if the involved error functions (2.9) for all edges are linear with regard to $\mathbf{x}$. In general, this is not the case. Nonetheless, optimization-based approaches force the output uncertainty to be Gaussian distributed. That is why the Jacobian needs to be used for the uncertainty propagation, as only linear functions produce Gaussian distributions from Gaussian input. That means, while the optimization provides a powerful method to compute a very likely solution, it fails to provide a reliable estimate of the uncertainty of the state parameters $\mathbf{x}$.



(a) Observation of the landmark $\mathbf{l}_1$ from pose $\mathbf{x}_1$.



(b) Robot odometry from pose $\mathbf{x}_1$ to $\mathbf{x}_2$.



(c) Several steps later.

Figure 2.4: Structure of the system information matrix $\mathbf{H}$ of the exemplary graph in Figure 4.4 (based on [18]). The graph is visualized on the left, and the system information matrix $\mathbf{H}$ is on the right. Non-zero blocks in $\mathbf{H}$ are colored blue, gray, or orange. All white cells represent zero blocks. Figure 2.4a only considers the first observation constraint. The concerning blocks are colored in $\mathbf{H}$. The effect of an odometry constraint is illustrated in Figure 2.4b. Odometry constraints populate the sub-diagonal blocks with non-zero entries. The system information matrix for the whole graph is shown in Figure 2.4c.

Outlier Treatment – Robust Kernels

Outliers in the data association in the front-end are not always avoidable. Already a few outliers are sufficient to let an optimization process diverge. The main problem caused by outliers is that they generate significant errors. Since the error we want to minimize stems from a quadratic objective function, the impact of outliers on the overall error may overshadow the error of inliers. As illustrated in Figure 2.5a, the larger the measurement error, the more significant its contribution to the overall error, so outliers will disproportionally influence the optimization.

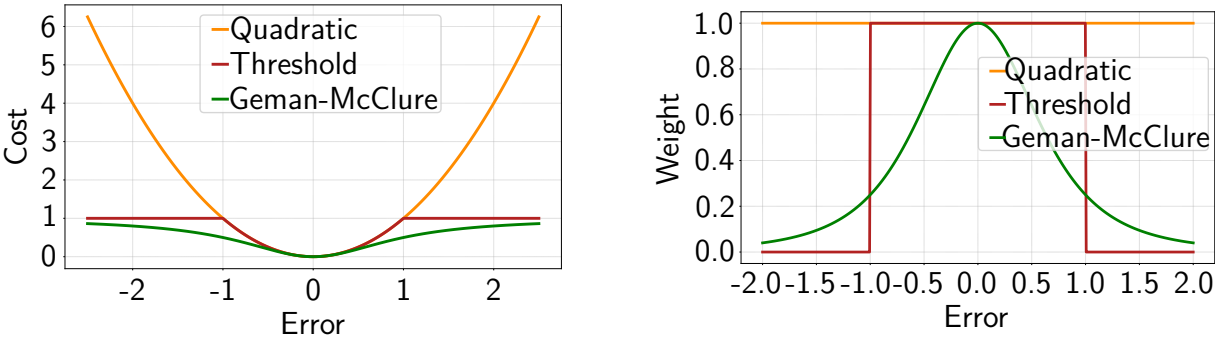
Hence, for optimization approaches, taking explicit care of outliers is vital. The literature shows two popular outlier treatments in the least squares context. On the one hand, Random Sample Consensus (RANSAC) approaches select the minimum number of measurements required to satisfy the model, counting the total number of measurements in agreement and repeating the process many times. The measurements are chosen randomly for each time. The largest consensus is considered the inlier set, while the remaining measurements are discarded as outliers. On the other hand, robust M-estimation is an alternative approach that addresses the outlier treatment by exchanging the least squares cost function introduced in (2.10) by a robust cost function (also called robust kernels) that decreases the influence of outliers. According to the literature, the second method has proven to be a good choice for robotics applications like SLAM and localization [25–29]. That is why we will stick to robust kernels to deal with outliers in the context of least squares optimization.

The core idea of robust M-estimation is to map the quadratic error (2.10) to a weighted error that down-weights the impact of large errors. The robust kernel determines the way how the error is down-weighted. In the literature, different types of robust kernel functions are suggested. In [25], a collection of robust kernels is presented and compared for different optimization problems corrupted by outliers. To gain a general understanding of what those kernel functions do, let us generally assume that a kernel function is described by

$$\hat{\mathbf{F}}_{ij} = \rho(\mathbf{F}_{ij}), \tag{2.27}$$

where $\mathbf{F}_{ij}$ is the quadratic error from (2.10) for the measurement $\mathbf{z}_{ij}$ connecting the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$. The kernel function ρ maps the quadratic error to $\hat{\mathbf{F}}_{ij}$. To reflect the weighted error in the optimization, typically, the information matrix Ω_{ij} is updated based on a weight function $w(\mathbf{F}_{ij})$ to obtain $\hat{\Omega}_{ij} = w(\mathbf{F}_{ij}) \cdot \Omega_{ij}$ which replaces the original information matrix Ω_{ij}. All the following calculations in the optimization procedure remain identical using the updated information matrix $\hat{\Omega}_{ij}$.

In this work, we will restrict ourselves to the Geman-McClure and Threshold error functions out of a large family of kernel functions offered in the literature (cf. [25]). Figure 2.5a depicts the resulting cost functions, while the weight of measurements with different errors in the optimization is reflected in Figure 2.5b. We also plot the classical quadratic error and weight without robustification for comparison.



(a) Robust cost functions.

(b) Robust weights.

Figure 2.5: Figure 2.5a visualizes the robust cost functions and Figure 2.5b shows the corresponding weights. The cost function determines the impact of the error on the optimization procedure. The weight function determines how the errors are weighted to obtain the corresponding cost function.

The Threshold cost function – sometimes also called saturated cost function – clips all errors that exceeded a predefined maximal error of t to a maximally permitted error σ and is described by

$$\rho_{\mathsf{T}}(\mathbf{F}_{ij}) = \begin{cases} \mathbf{F}_{ij} \text{ for } \mathbf{F}_{ij} \leq t^2, \\ \sigma^2 \text{ for } \mathbf{F}_{ij} > t^2 \end{cases} . \tag{2.28}$$

However, the saturated cost function typically sets the weight to update the information matrix to

$$w_{\mathsf{T}}(\mathbf{F}_{ij}) = \begin{cases} 1 \text{ for } \mathbf{F}_{ij} \leq t^2, \\ 0 \text{ for } \mathbf{F}_{ij} > t^2 \end{cases} . \tag{2.29}$$

Accordingly, all those measurements that exceed the error threshold are discarded from the optimization as they do not have any impact due to zero weight. We illustrate this in Figure 2.5b.

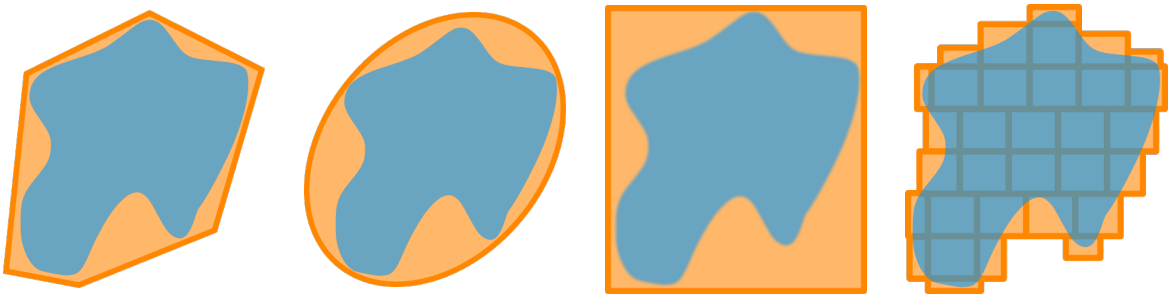
The Geman-McClure kernel function [30] is

$$\rho_{\mathsf{GM}}(\mathbf{F}_{ij}) = \frac{\mathbf{F}_{ij}}{\sigma + \mathbf{F}_{ij}}, \tag{2.30}$$

and is visualized in Figure 2.5a. In this expression, σ is a parameter that defines the shape of the cost function. This cost function weights larger errors lower than smaller errors as indicated in Figure 2.5b. As a result, the impact of larger errors on the optimization result reduces. The weight function is described by

$$w_{\mathsf{GM}}(\mathbf{F}_{ij}) = \frac{\sigma^2}{(\sigma + \mathbf{F}_{ij})^2}. \tag{2.31}$$

The weight function is shown in Figure 2.5b and indicates that the larger the error, the closer the weight converges to zero so that the influence on the optimization of measurements fraught with larger error decreases. Summing up, robust kernels deal with outliers by down-weighting the impact of large errors in the overall objective function by manipulating the information matrix in the way sketched above.



(a) Polyhedral. (b) Ellipsoidal. (c) Interval. (d) Subpaving.

Figure 2.6: Set membership approaches to enclose an arbitrary set $\mathcal{S}$. $\mathcal{S}$ is colored blue and the enclosure is colored orange.

2.2 Interval Analysis

Since the first proof of the existence of irrational numbers by the Pythagorean Hippasus of Metapontum [31], the consideration of irrational numbers has raised the question of their decimal representation. Using interval bounds to approximate unknown but bounded values has been a widely and extensively used tool in mathematics. The well-known Greek mathematician Archimedes already provided a reasonably small enclosure of the irrational number π [32]. By approximating the circle by an inner and outer 96-sided polygon, Archimedes concluded $\frac{223}{71} < \pi < \frac{22}{7}$.

With the advent of numerical computation, interval analysis gained further popularity. Due to finite floating-point computations within computers, the representation of real numbers of infinite precision inevitably leads to an approximation error. This prompted scientists to tackle the question how to model the error, leading to the first notable book on interval analysis in 1966, authored by Ramon E. Moore [33]. This work paved the way for the application of interval analysis in various disciplines such as solving interval equation systems [34, 35], advanced digital computer arithmetic [36, 37], recursive state estimation [38, 39], numerical error propapgation [40, 41], and global optimization [42–45]. Although interval analysis has been broadly applied in many different fields, this thesis will focus on the uncertainty modeling of physical quantities. In robotics, we can determine bounds for measurement errors and perform reliable computations. Employing interval analysis, the measurement uncertainties can be reliably propagated to the involved physical quantities we seek to deduce from the uncertain measurements.

Interval analysis is part of a large family of set-membership methods. Where interval analysis applies intervals to enclose a set, other geometrical structures such as polygons [46], ellipsoids [38, 47] or zonotopes [48] may also be used. Figure 2.6 depicts different types of set-membership approximations of an arbitrary set $\mathcal{S}$. Note that Figure 2.6d depicts a special type of interval enclosure, where the set is represented by multiple subsets, which are intervals. We call such a representation a subpaving. All set-membership approaches overestimate $\mathcal{S}$. We call this approximation of $\mathcal{S}$ the *outer approximation* since no region of $\mathcal{S}$ is outside of the approximation. We qualify an approach to be better the less the approximation overestimates the set we want to represent. As a result, the polyhedral and ellipsoidal enclosure seem to better approximate $\mathcal{S}$ compared to the interval enclosure in Figure 2.6c. However, the computation

with ellipsoidal and polyhedral sets can become convoluted. To solve robotic tasks, intervals have proven to be an appropriate representation since efficient computations are possible.

Applying interval analysis to model the uncertainty of measurements and states in the context of robotics implies fundamentally different assumptions compared to classical probabilistic approaches. We must define a probability density function for the measurements to apply a probabilistic approach. As explained in the previous section, a normal distribution of the measurements is typically assumed. By propagating the probability densities via the state equations to the density functions that describe the states, we obtain probabilities for individual states. As discussed in the previous section, the exact propagation procedure is computationally infeasible, due to which unavoidable approximations affect the propagation and potentially distort the states' probability density functions to an unacceptable extent.

In contrast to probabilistic approaches, interval approaches assume the measurement error to be bounded. Those bounds are propagated via the state equations to the set of feasible states represented by intervals. That means while interval approaches introduce the assumption that the selected bounds enclose the correct measurement, probabilistic approaches define probabilities on a set of measurements. The interval approach provides a set of feasible solutions by applying the propagation tools, while the probabilistic approach provides probabilities for individual solutions.

Consequently, classical probabilistic and interval analysis approaches model sensor errors differently since they make different assumptions about the real world. Probabilistic approaches assume the error distribution to be known, but interval-based approaches assume the error can be bounded reliably so that the estimate encloses the correct value. However, none of those assumptions are universally correct for the real world. The selection of the appropriate error model always depends on what we want to compute and which assumptions fit the real world best.

In the following, the notions and definitions are taken from Luc Jaulin's book [49], Simon Rohou's Ph.D. thesis [50], and Raphael Voges' Ph.D. thesis [51]. For the numerical interval computations, we employ the publicly available *IBEX* library [52].

2.2.1 Basic Notions and Operations

An interval $[x]$ is a closed and connected subset of $\mathbb{R}$. $\mathbb{IR}$ denotes the set of all intervals. The interval $[x]$ is defined by

$$[x] = [\underline{x}, \overline{x}] = \{\, x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x} \,\}, \tag{2.32}$$

where $\underline{x}$ and $\overline{x}$ set the lower and upper bounds of $[x]$ as illustrated in Figure 2.7. The lower as well as the upper bound can be infinite. In the case of $\underline{x} = \overline{x}$, the interval $[x]$ is said *degenerate*. Consequently, any real number can generally be considered a degenerate interval. The same applies to the empty set $\varnothing$, which denotes the absence of a solution to our problems.

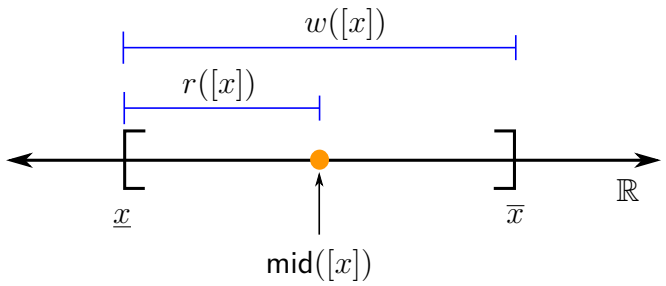


Figure 2.7: Definition of an interval.

The actual but unknown value we aim to estimate – bounded by its interval estimation $[x]$ – will be denoted by x^*. In the interval literature, the uncertainty of the enclosure $[x]$ of the desired value x^* is given by the interval's width

$$w([x]) = \overline{x} - \underline{x}. \tag{2.33}$$

To generate a commensurate metric that allows for direct comparison between interval analysis and confidence intervals typical for probability theory, we will use the radius

$$r([x]) = \frac{\overline{x} - \underline{x}}{2} \tag{2.34}$$

as a measure of uncertainty of x^*. This enables the direct comparison with the standard deviation in case of Gaussian uncertainties.

Moreover, the midpoint of an interval $[x]$ is defined by

$$\mathrm{mid}([x]) = \frac{\overline{x} + \underline{x}}{2}. \tag{2.35}$$

The midpoint may serve as an initial approximation of x^*.

Example 2.2.1. In the following, exemplary intervals, their radius, and their midpoint are specified:

$[x]$	$r([x])$	$\mathrm{mid}([x])$
$[-4, 6]$	5	1
$[5]$	0	5
$\varnothing$	undefined	undefined
$[-\infty, \infty]$	∞	undefined
$[5, \infty]$	undefined	undefined

2.2.1.1 Operation on Sets

All operations defined in set theory are applicable to intervals. The intersection between two intervals $[x]$ and $[y]$ is defined by

$$[x] \cap [y] = \{\, z \in \mathbb{R} \mid z \in [x] \text{ and } z \in [y] \,\}. \tag{2.36}$$

The intersection of two intervals always results in a new interval.

The union is denoted by

$$[x] \cup [y] = \{\, z \in \mathbb{R} \mid z \in [x] \text{ or } z \in [y] \,\}. \tag{2.37}$$

In contrast to the intersection, the union of two intervals is not necessarily an interval, as the resulting set may not fulfill the connectedness property.

Example 2.2.2. $[5, 7] \cup [9, 10]$ is not an interval since the result is not connected.

To overcome this problem, we need to introduce the notion of an interval hull. We denote the interval hull to the set $\mathbb{X} \subset \mathbb{R}$ by $[\mathbb{X}]$, which represents the smallest interval containing all values of $\mathbb{X}$. Hence, the interval hull over $[x] \cup [y]$ represents the smallest interval containing $[x]$ and $[y]$. As a consequence, $[x] \cup [y] \subset [[x] \cup [y]]$ holds. Note that the hull increases pessimism in the estimate, as the hull may contain parts that are not part of the union.

Example 2.2.3. Exemplary operations are listed in the following:

- $[3, 5] \cap [4, 6] = [4, 5]$,
- $[1, 2] \cap [4, 6] = \varnothing$,
- $[5, 7] \sqcup [9, 10] = [5, 10]$,
- $[x] \cap \varnothing = \varnothing$,
- $[x] \sqcup [-\infty, \infty] = [-\infty, \infty]$.

2.2.1.2 Interval Computations

Besides set-theoretic operations, interval analysis also extends real arithmetic operators to intervals. Let $\diamond \in \{+, -, \cdot, /\}$ be one of the for classical operators. Applying an operator to the two intervals $[x]$ and $[y]$ results in

$$[x] \diamond [y] = [\{\, x \diamond y \in \mathbb{R} \mid x \in [x], y \in [y] \,\}]. \tag{2.38}$$

Hence, the result defines the smallest interval that contains all results for $x \diamond y$ for every $x \in [x]$ and $y \in [y]$. As intervals represent a connected set, the computation of the operations mentioned above only needs to consider the bounds. For instance, addition can be performed by

$$[x] + [y] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]. \tag{2.39}$$

Definitions for the other classical operators can be found in [49].

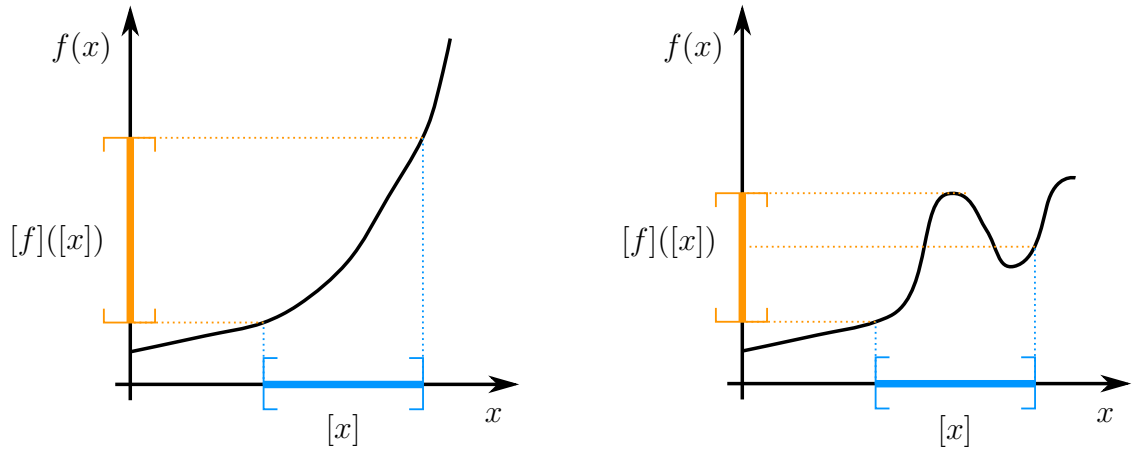
Example 2.2.4.

- $[3, 5] + [4, 6] = [7, 11]$,
- $[1, 2] \diamond \varnothing = \varnothing$.

The extension of the basic operators from $\mathbb{R}$ to $\mathbb{IR}$ also leads to different properties of the operators. For example the distributive law $x \cdot (y + z) = x \cdot y + x \cdot z$ does not apply to intervals since

$$[x] \cdot ([y] + [z]) \subset [x] \cdot [y] + [x] \cdot [z] \tag{2.40}$$

holds. Further, the subtraction of an element by itself $[x] - [x] \neq [0]$ does not result in the neutral element of addition in general. Hence, dealing with intervals requires careful



(a) Monotonic function. (b) Non-monotonic function.

Figure 2.8: Interval computation for monotonic and non-monotonic functions.

examination of the related equations, and often it is important to simplify equations to obtain tighter intervals.

The arithmetic extension includes elementary functions such as $\sin$, $\cos$, and $\exp$. Sometimes the image of an interval $[x]$ belonging to a function f is not necessarily an interval – as is the case for discontinuous functions. Therefore, the evaluation of $f([x])$ – denoted by $[f]([x])$ – is defined by the smallest interval enclosure containing all the images of $[x]$ through f:

$$[f]([x]) = [\{\, f(x) \mid x \in [x] \,\}]. \tag{2.41}$$

If f is monotonic, the interval evaluation simplifies to evaluating the upper and lower bound as illustrated in Figure 2.8a. Since the property of a monotonic function is that it only increases or decreases for an increasing argument, the bounds of the argument $[x]$ are sufficient to determine the resulting interval $[f]([x])$. For instance, $\exp$ is such a monotonic function for which

$$[\exp]([x]) = [\exp(\underline{x}), \exp(\overline{x})] \tag{2.42}$$

holds. However, treating non-monotonic functions is more complicated, as illustrated in Figure 2.8b. For functions like, for example, $\sin$ and $\cos$, the evaluation of the bounds only is insufficient. Special algorithms must be considered to compute the image of such functions [49].

2.2.1.3 Interval vectors

We define the Cartesian product of n intervals

$$[\mathbf{x}] = [x_1] \times ... \times [x_n] = \begin{pmatrix} [x_1] & ... & [x_n] \end{pmatrix}^T \tag{2.43}$$

an interval vector – we also call it box – defining a subset of $\mathbb{R}^n$. The set of all interval vectors is described by $\mathbb{IR}^n$. An interval vector represents an axis-aligned box, where the i-th component is the interval that results from the projection of the box to the i-th axis. In Figure 2.9, a two dimensional box $[\mathbf{x}] \in \mathbb{IR}^2$ is illustrated. The already introduced operations on intervals are

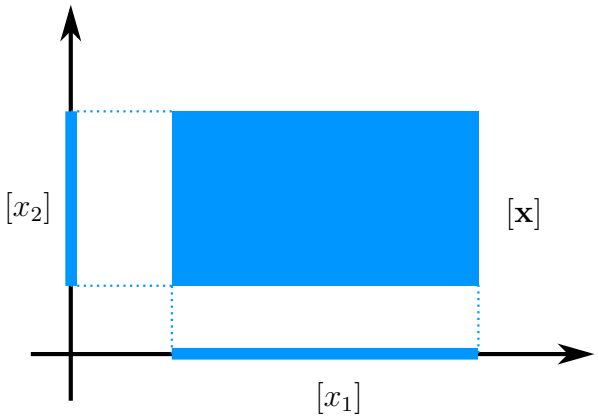


Figure 2.9: A two dimensional interval vector $[\mathbf{x}] = [x_1] \times [x_2]$ and the projected intervals.

extended to interval vectors by performing the corresponding operation on each component of the interval vector. Similarly, interval matrices are constructed in analogy to the interval vectors.

2.2.1.4 Inclusion functions

Let us consider an arbitrary function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. If we compute the image set $\mathbf{f}([\mathbf{x}])$ for every possible value in the input box $[\mathbf{x}]$, $\mathbf{f}([\mathbf{x}])$ does not necessarily result into a box. The image set can have any arbitrary shape, contain disconnected subsets, and may also have holes, as illustrated in Figure 2.10. The representation and computation of such complicated sets can become computationally expensive. Therefore, we introduce the notion of inclusion functions that approximate the complicated image sets by enclosing boxes.

We define an inclusion function $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^m$ to enclose the image of $[\mathbf{x}]$ by $\mathbf{f}$ in a box such that $\forall [\mathbf{x}] \in \mathbb{IR}^n : \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}])$. The inclusion function aims to provide the interval enclosure reasonably fast.

An inclusion function $[\mathbf{f}]$ is *minimal* if $\forall [\mathbf{x}], [\mathbf{f}]([\mathbf{x}])$ is the smallest box containing $\mathbf{f}([\mathbf{x}])$. We denote the unique minimal inclusion function as $[\mathbf{f}]^*$, as also illustrated in Figure 2.10. Any non-minimal inclusion function is dubbed *pessimistic* since it overestimates the image set (cf. Figure 2.10). Furthermore, we define an $[\mathbf{f}]$ to be *thin* if the image of any degenerate interval vector $[\mathbf{x}] = \mathbf{x}$ is also degenerate. $[\mathbf{f}]$ is said *inclusion monotonic* if $[\mathbf{x}] \subset [\mathbf{y}] \Rightarrow [\mathbf{f}]([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{y}])$.

Natural Inclusion Function

When the function $\mathbf{f}$ is composed of elementary functions such as $\sin$, $\cos$, $\sqrt{(\cdot)}$, and operators $+$, $-$, $\cdot$, $/$, the simplest way to obtain an inclusion function is to replace each variable x_i by its interval representation $[x_i]$ and each function and operator by their interval counterpart defined on $\mathbb{IR}$. We call a function $[\mathbf{f}]$ that is obtained this way a *natural inclusion function* of $\mathbf{f}$.

A natural inclusion function is always thin and inclusion monotonic by construction. In general, the natural inclusion function is not necessarily minimal because of possible dependencies between variables and the wrapping effect (cf. Section 2.2.3). Nonetheless, if a function is

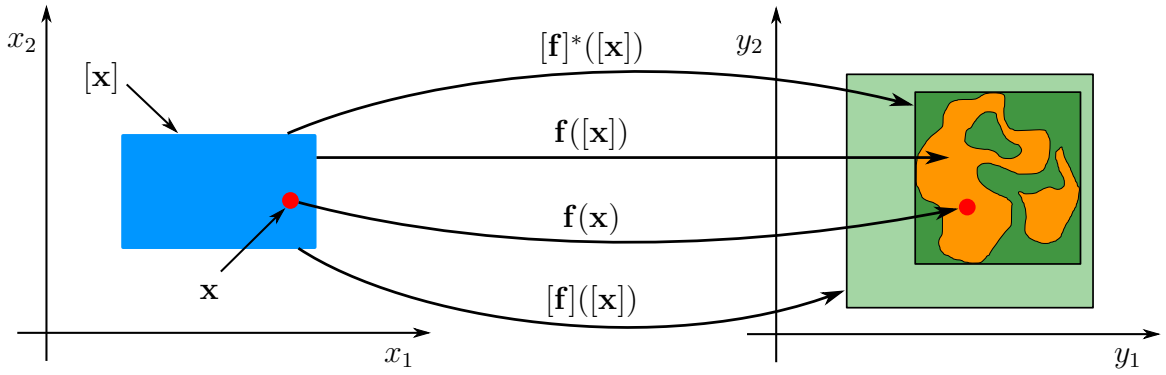


Figure 2.10: Illustration of an arbitrary function $\mathbf{f}$ that maps a box $[\mathbf{x}]$ (blue) to a disconnected image set $\mathbf{f}([\mathbf{x}])$ colored in orange. While $[\mathbf{f}]([\mathbf{x}])$ is an arbitrary inclusion function to $\mathbf{f}$, $[\mathbf{f}]^*([\mathbf{x}])$ is the minimal inclusion function.

only composed of continuous operators and functions, and if each of the variables only appears once in the equations, the natural inclusion function is minimal.

2.2.2 Constraint Satisfaction Problems (CSP)

CSPs formulate a family of problems in which we seek to find from an initial set $\mathbb{X}$ a subset $\mathbb{S}$ that contains all values that satisfy all constraints defined in the CSP. Consider n variables $x_i \in \mathbb{R}$ with $1 \leq i \leq n$, bound by m constraints of the form

$$f_j(x_1, ..., x_n) = 0,\ j \in \{1, ..., m\}. \tag{2.44}$$

Each variable x_i belongs to an interval domain $[x_i]$. For simplicity of notation we assemble all variables to vector $\mathbf{x} = \begin{pmatrix} x_1 & ... & x_n \end{pmatrix}^T$. Hence, the prior domain for $\mathbf{x}$ is the interval vector $[\mathbf{x}]$.

Furthermore, we arrange all constraints $f_j(\mathbf{x}) = 0$ to vector form so that our constraints can be written as $\mathbf{f}(\mathbf{x}) = 0$. This corresponds to the CSP $\mathcal{H}$ which we formulate as

$$\mathcal{H}\ :\ (\mathbf{f}(\mathbf{x}) = \mathbf{0}, \mathbf{x} \in [\mathbf{x}]). \tag{2.45}$$

The solution set $\mathbb{S}$ is the subset among the initial domain $[\mathbf{x}]$ that satisfies all constraints defined in $\mathcal{H}$.

$$\mathbb{S} = \{\mathbf{x} = [\mathbf{x}] \,|\, \mathbf{f}(\mathbf{x}) = \mathbf{0}\}. \tag{2.46}$$

Example 2.2.5. An exemplary CSP is given by its constraints and the initial domains of the variable:

$$\mathcal{H}\ :\ \left(\begin{array}{l} \mathbf{f}(\mathbf{x}) = \begin{cases} (x_1 - 3.5)^2 + (x_2 - 3.5)^2 - r_1^2 = 0 \\ (x_1 - 7)^2 + (x_2 - 4.5)^2 - r_2^2 = 0 \\ x_1 - x_2 + b = 0 \end{cases} \\ [x_1] = [1, 9],\ [x_2] = [0.5, 8],\ [r_1] = [2, 2.5],\ [r_2] = [3.5, 4.5],\ [b] = [1.5, 2.5] \end{array} \right). \tag{2.47}$$

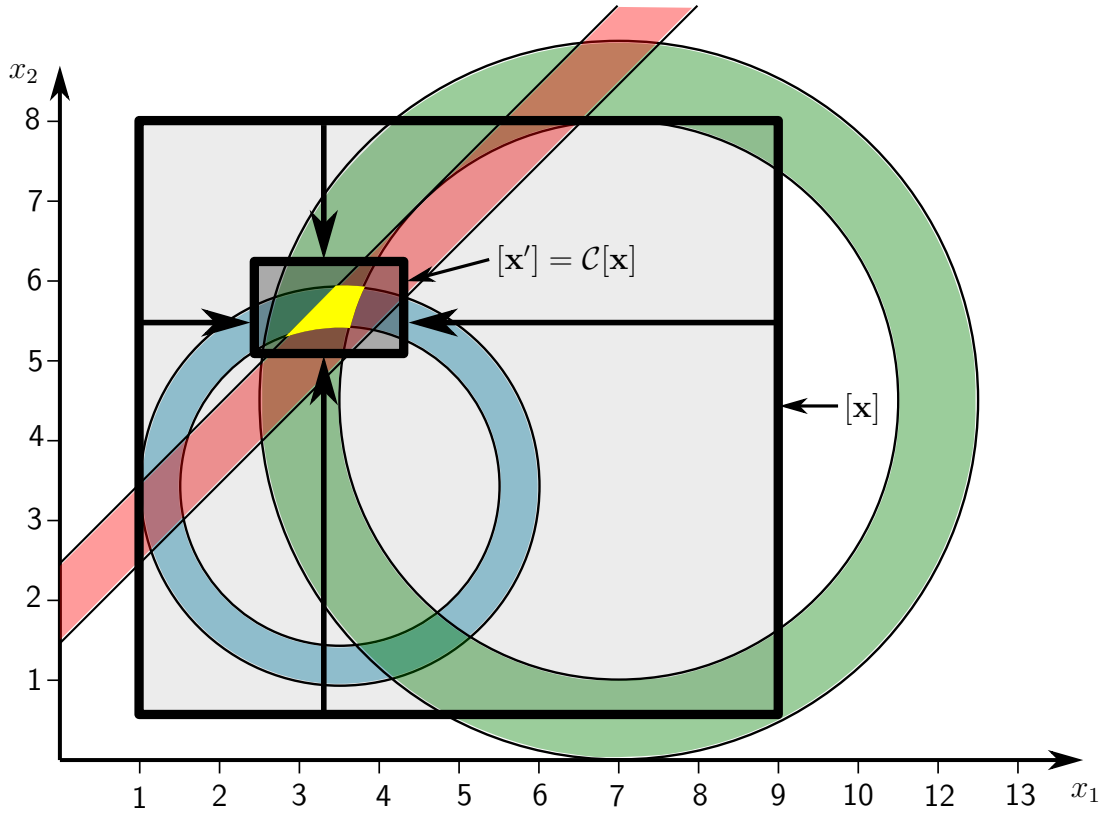


Figure 2.11: Illustration of how a contractor works based on the Example 2.2.5. Each constraint of the CSP $\mathcal{H}$ is visualized. The first and second constraints are circles, while the third is a stripe. The exact solution set is the intersection of all geometries and is highlighted yellow. The contractor $\mathcal{C}$ contracts the initial set $[\mathbf{x}]$ to a smaller subset $[\mathbf{x}']$.

Figure 2.11 visualizes the CSP. The first and second constraints represent two ring areas. The first constraint corresponds to the smaller blue area, and a green area visualizes the second constraint. The third constraint is a stripe in the 2D space visualized in red. The solution set of the CSP is the intersection highlighted in yellow. The initial domain for $[\mathbf{x}]$ is represented in Figure 2.11 by the large black-bordered box.

Unfortunately, finding the solution set $\mathbb{S}$ of a CSP is generally NP-hard. To overcome this difficulty, we use contractors to approximate the solution set with a reasonably tight enclosure of $\mathbb{S}$ in polynomial time.

2.2.2.1 Contractors

A contractor $\mathcal{C}$ reduces the initially large domain $[\mathbf{x}]$ to a smaller domain $[\mathbf{x}'] \subset [\mathbf{x}]$ such that the solution set remains unchanged so that $\mathbb{S} \subset [\mathbf{x}']$ holds. In other words, only those subsets of the initial domain $[\mathbf{x}]$ are dismissed by the contractor that are not part of the solution set $\mathbb{S}$. No bisections of the domains are allowed to maintain a polynomial time and space complexity for contractors. In Figure 2.11, the general idea of a contractor is visualized for the CSP introduced in Example 2.2.5. The contractor $\mathcal{C}$ takes the large initial interval box $[\mathbf{x}]$ as input and contracts it to a smaller box $[\mathbf{x}']$, that still contains the whole solution set, which is highlighted yellow.

Formally speaking, a contractor is an operator $\mathcal{C} : \mathbb{IR}^n \to \mathbb{IR}^n$, which is associated with a constraint and returns a box $\mathcal{C}([\mathbf{x}]) \subseteq [\mathbf{x}]$ without removing any value that is consistent with the constraint. As such, a contractor is characterized by two fundamental properties:

1. **Contraction**: $\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathcal{C}([\mathbf{x}]) \subseteq [\mathbf{x}]$,
2. **Consistency**: $(\mathbf{x} \in [\mathbf{x}], \mathcal{C}(\{\mathbf{x}\}) = \{\mathbf{x}\}) \Rightarrow \mathbf{x} \in \mathcal{C}([\mathbf{x}])$.

While the first property ensures that a contractor can only reduce but never enlarge a box, the second property guarantees that the contractor never loses a point that satisfies the constraint. Accordingly, a contractor can be applied to a box as often as it is deemed advisable without losing parts of the solution subsets enclosed in the initial domain.

Often one is interested in finding the minimal contractor that manages to contract the box $[\mathbf{x}]$ to the smallest box containing the solution set. In Figure 2.11, a minimal contractor provides the smallest possible axis-aligned box that encloses the yellow region. Finding such a minimal contractor is not trivial. However, if a contractor is found to be minimal, it allows to efficiently determine the enclosure of the solution set by just applying it once. For the non-minimal case, it may be required to iterate the contraction process multiple times.

Forward-Backward Contractor

One of the most important and widely used contractors is the forward-backward contractor $\mathcal{C}_{\uparrow\downarrow}$. It exploits that constraints are formulated as a sequence of operations involving elementary operators and functions. Decomposing those constraints into their most basic components and considering them for contraction in isolation, the $\mathcal{C}_{\uparrow\downarrow}$ is suitable to perform contractions for non-linear constraints. We illustrate the construction of a forward-backward contractor based on the following example.

Example 2.2.6. Let us consider the following CSP.

$$\mathcal{H} : \begin{pmatrix} f(\mathbf{x}) = x_1 \cdot \exp(x_2) + x_3^2 = 0 \\ x_1 \in [x_1],\, x_2 \in [x_2],\, x_3 \in [x_3] \end{pmatrix}. \tag{2.48}$$

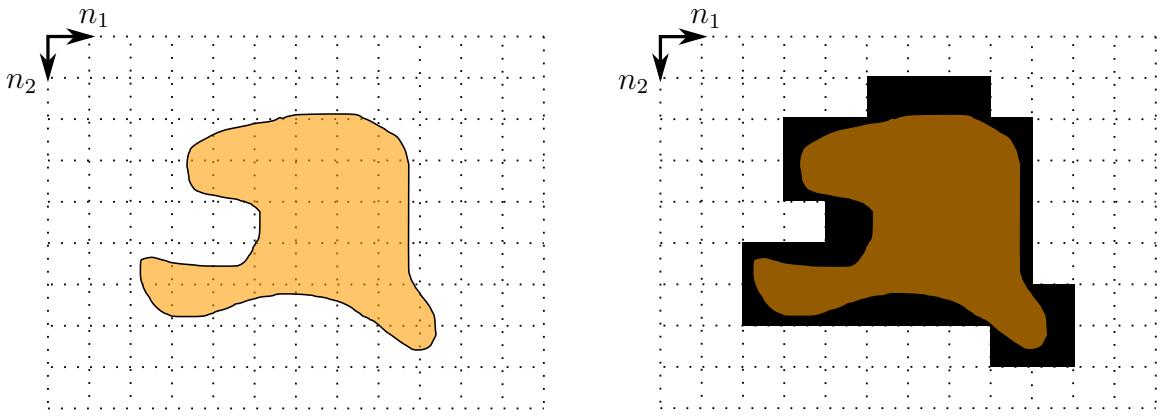
In the forward pass, we split the function $y = f(\mathbf{x})$ into the sequence of operations. Therefore, we need to introduce intermediate variables a_i for $i > 0$ and utilize the interval inclusion functions for the operators:

1. $[a_1] = \exp([x_2])$;
2. $[a_2] = [x_1] \cdot [a_1]$;
3. $[a_3] = [x_3]^2$;
4. $[y] = [a_2] + [a_3]$.

Since our constraint is $f(\mathbf{x}) = y = 0$, we can add the further step

5. $[y] = [y] \cap [0]$.

If this process step leads $[y]$ to be empty, the solution set of the CSP has to be empty. If not, $[y]$ is replaced by $[0]$. The final step is the backward pass, where all associated domains are updated utilizing the inverse primitive operators and functions.



(a) Complex shape that is hard to parametrize. (b) Approximation of the shape by a binary image.

Figure 2.12: Approximation of complex shapes by a binary image that denotes a pixel value of 1 if the respective region is part of the shape. The corresponding pixel is colored black. Those pixels not part of the shape are colored white and have the value 0.

6. $[a_2] = [a_2] \cap ([y] - [a_3])$; $[a_3] = [a_3] \cap ([y] - [a_2])$;
7. $[x_3] = [x_3] \cap \sqrt{[a_3]}$;
8. $[x_1] = [x_1] \cap \frac{[a_2]}{[a_1]}$; $[a_1] = [a_1] \cap \frac{[a_2]}{[x_1]}$;
9. $[x_2] = [x_2] \cap \log([a_1])$;

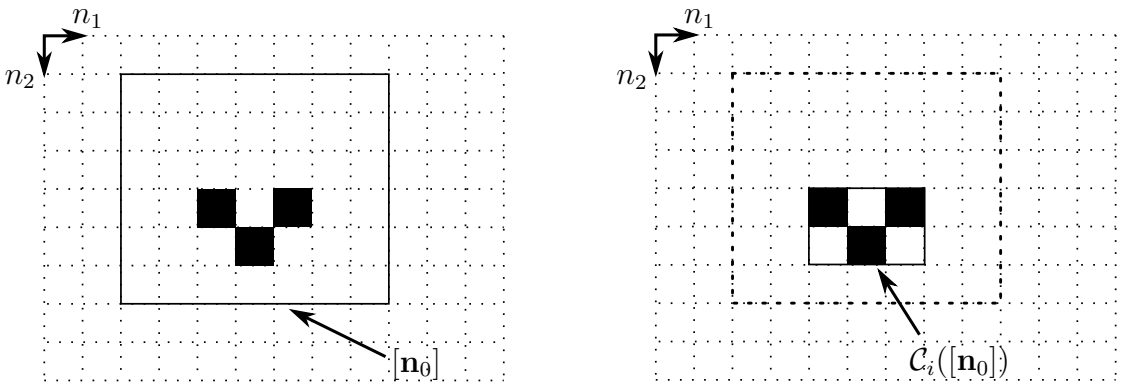
Steps 6 and 8 consist of two operations since the backward propagation is performed for binary operators.

Generally, the forward-backward contractor is not minimal and must be applied multiple times to obtain a reasonably small box. However, it is minimal if each variable only appears once in all constraints. This said, the forward-backward contractor we introduce in Example 2.2.6 is minimal.

Image Contractor

The forward-backward contractor is well applicable for those cases where we can describe the solution set by a set of equations. However, describing an unstructured set $\mathbb{S}$ as depicted in Figure 2.12a by a set of equations may become inconvenient. A parametric representation seems inappropriate, especially if we think of arbitrary building maps of large cities. To deal with such unstructured shapes, Sliwka et al. first introduced in [53] the image contractor. Desrochers [54] used the contractor to solve localization tasks in unstructured environments. Here we will stick to Desrochers' notation in [54].

We need to describe the region by a binary image to apply the image contractor. Hence, we need to approximate the region $\mathbb{S}$ by a set of pixels structured in a raster. The binary image encodes for pixels with a non-empty intersection with $\mathbb{S}$ with the value 1, 0 otherwise. The approximated binary image representation of the orange set in Figure 2.12a is colored black in Figure 2.12b. Note that the approximation error depends on the raster size – the more pixels the binary image contains, the more accurate the approximation will be. Furthermore, the binary image representation of $\mathbb{S}$ always overestimates the region since we set all those pixels



(a) Initial box $[\mathbf{n}_0]$. (b) Result after applying the image contractor.

Figure 2.13: Illustration of the image contractor based on [54].

to 1 with a non-empty intersection with $\mathbb{S}$. So by construction, we ensure that we never lose any subsets of $\mathbb{S}$.

The image contractor uses the so-called integral image of the binary image constructed from the set $\mathbb{S}$ as explained above. The integral image is widely used in computer vision [54]. It represents and stores for each pixel the sum of pixel values located between the image's top-left corner and the considered pixel. As a result, given a binary image i the integral image I for the pixel position (n_1, n_2) is defined by

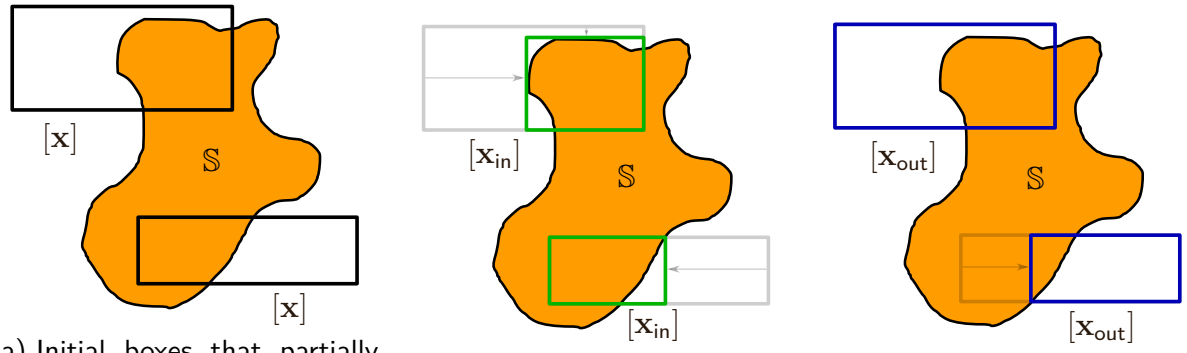
$$I(n_1, n_2) = \sum_{n_1' \leq n_1,\, n_2' \leq n_2} i(n_1', n_2') \tag{2.49}$$

The integral image has to be computed beforehand based on the binary image. Utilizing the integral image, the number of 1-valued pixels in any rectangular region can be efficiently computed by just four operations. Let $[\mathbf{n}] = \begin{pmatrix}[n_1] & [n_2]\end{pmatrix}^T$ be a box aligned on the grid of a binary image. We define the function ϕ, which returns the number of 1-valued pixels in a given box $[\mathbf{n}]$ by

$$\phi([\mathbf{n}]) = I(\underline{n_1}, \underline{n_2}) + I(\overline{n_1}, \underline{n_2}) - I(\overline{n_1}, \overline{n_2}) - I(\underline{n_1}, \overline{n_2}). \tag{2.50}$$

For example the box $[\mathbf{n}_0]$ in Figure 2.13a contains three black pixels – that means $\phi([\mathbf{n}_0]) = 3$. The aim of the image contractor $\mathcal{C}_{\text{im}}$ is to find the smallest box $[\mathbf{n}'] = \mathcal{C}_{\text{im}}([\mathbf{n}_0])$ in the binary image, that contains the same number of 1-valued pixels – which means that $\phi([\mathbf{n}_0]) = \phi([\mathbf{n}'])$ holds. This is illustrated in Figure 2.13b. The algorithm of the image contractor $\mathcal{C}_{\text{im}}([\mathbf{n}]) = [\mathbf{m}]$ for $[\mathbf{n}] = \begin{pmatrix}[n_1] & [n_2]\end{pmatrix}^T$ and $[\mathbf{m}] = \begin{pmatrix}[m_1] & [m_2]\end{pmatrix}^T$ only consists of two $\min$ and two $\max$ operations:

1. $\underline{m_1} = \max\left(x \in [n_1],\, \phi\begin{pmatrix}[\underline{n_1}, x] \\ [n_2]\end{pmatrix} = 0\right)$;
2. $\overline{m_1} = \min\left(x \in [n_1],\, \phi\begin{pmatrix}[x, \overline{n_1}] \\ [n_2]\end{pmatrix} = 0\right)$;
3. $\underline{m_2} = \max\left(x \in [n_2],\, \phi\begin{pmatrix}[m_1] \\ [\underline{n_2}, x]\end{pmatrix} = 0\right)$;
4. $\overline{m_2} = \min\left(x \in [n_2],\, \phi\begin{pmatrix}[m_1] \\ [x, \overline{n_2}]\end{pmatrix} = 0\right)$.



(a) Initial boxes that partially overlap with $\mathbb{S}$. The boxes are colored black, and $\mathbb{S}$ is orange.

(b) Outer approximation $\mathcal{C}_{\mathbb{S}}([\mathbf{x}]) = [\mathbf{x}_{\text{in}}]$ of the subsets inside $\mathbb{S}$.

(c) Outer approximation $\mathcal{C}_{\overline{\mathbb{S}}}([\mathbf{x}]) = [\mathbf{x}_{\text{out}}]$ of the subsets outside $\mathbb{S}$.

Figure 2.14: Illustration of a separator applied on different initial boxes in Figure 2.14a. The contraction results for the set $\mathbb{S}$ are shown in Figure 2.14b, while the result concerning $\overline{\mathbb{S}}$ is shown in Figure 2.14c.

The first two operations determine the interval $[m_1]$ that defines the horizontal size of the box. While the first operation determines the maximal lower bound $x \in [n_1]$ so that the box $\left([\underline{n_1}, x] \quad [n_2]\right)^T$ does not contain any 1-valued pixels, the second operator computes the minimal upper bound $x \in [n_1]$ so that the box $\left([x, \overline{n_1}] \quad [n_2]\right)^T$ also does not contain any 1-valued pixels. By setting the bounds for $[m_1]$, we ensure that none of the 1-valued pixels initially contained in $[\mathbf{n}]$ will be lost. The same operations are applied on the row interval $[m_2]$.

The efficiency of the image contractor mainly relies on the implementation of the $\min$ and $\max$ operators. We implemented the operators using dichotomy, which yields logarithmic complexity.

2.2.2.2 Separators

A contractor that is consistent to a set $\mathbb{S}$ replaces an input box $[\mathbf{x}]$ by a smaller box $[\mathbf{x}']$ so that all of the initially contained subsets of $\mathbb{S}$ in $[\mathbf{x}]$ are preserved in $[\mathbf{x}']$. So by construction, a contractor always provides the outer approximation of the set $\mathbb{S}$. Consequently, the systematic overestimation of $\mathbb{S}$ is the flipside of being sure never to lose parts of $\mathbb{S}$. However, for some applications, having the outer and inner approximation of the set $\mathbb{S}$ is useful. However, contrary to the outer approximation, we do not directly compute the inner approximation. Instead, we invert this problem to an outer approximation problem by contracting the initial box to the complementary set $\overline{\mathbb{S}}$. This approach is handy since we can compute the outer approximation using a contractor. Summing up, we apply two complementary contractors $\mathcal{C}_{\mathbb{S}}$ and $\mathcal{C}_{\overline{\mathbb{S}}}$ on an initial box $[\mathbf{x}]$ so that we obtain $\mathcal{C}_{\mathbb{S}}([\mathbf{x}]) = [\mathbf{x}_{\text{in}}]$ and $\mathcal{C}_{\overline{\mathbb{S}}}([\mathbf{x}]) = [\mathbf{x}_{\text{out}}]$, where $[\mathbf{x}_{\text{in}}]$ defines the outer approximation of the subsets inside $\mathbb{S}$ and $[\mathbf{x}_{\text{out}}]$ is the outer approximation of the subsets outside $\mathbb{S}$. We illustrate the complementary sets in Figure 2.14.

Following the terminology in [54], we call the operator that separates an initial box into two complementary subsets a separator. Thus, a separator $\mathcal{S}$ associated to a set $\mathbb{S}$ defines the operation

$$\mathcal{S} : \mathbb{IR}^n \to \mathbb{IR}^n \times \mathbb{IR}^n \tag{2.51}$$

$$[\mathbf{x}] \to ([\mathbf{x}_{\text{in}}], [\mathbf{x}_{\text{out}}]). \tag{2.52}$$

The separator has the following properties

$$\begin{array}{ll} \text{(i)} & [\mathbf{x}] = [\mathbf{x}_{\text{in}}] \cup [\mathbf{x}_{\text{out}}], \\ \text{(ii)} & [\mathbf{x}_{\text{in}}] \cap \mathbb{S} = [\mathbf{x}] \cap \mathbb{S}, \\ \text{(iii)} & [\mathbf{x}_{\text{out}}] \cap \overline{\mathbb{S}} = [\mathbf{x}] \cap \overline{\mathbb{S}}. \end{array} \tag{2.53}$$

Note that the first property expresses the separation property, stating that the two sets are complementary. The second property defines that $[\mathbf{x}_{\text{in}}]$ does not lose any subsets of $\mathbb{S}$ that were already considered in $[\mathbf{x}]$. In analogy, the third property describes that $[\mathbf{x}_{\text{out}}]$ does not lose any subsets of $\overline{\mathbb{S}}$ that were already considered in $[\mathbf{x}]$. Due to the separator's properties, we can also define it as a pair of contractors $\{\mathcal{C}_{\mathbb{S}}, \mathcal{C}_{\overline{\mathbb{S}}}\}$ such that for all $[\mathbf{x}] \in \mathbb{IR}^n$ the complementarity property

$$\mathcal{C}_{\mathbb{S}}([\mathbf{x}]) \cup \mathcal{C}_{\overline{\mathbb{S}}}([\mathbf{x}]) = [\mathbf{x}] \tag{2.54}$$

holds. A separator that is consistent to the set $\mathbb{S}$ is minimal if and only if the two contractors $\mathcal{C}_{\mathbb{S}}$ and $\mathcal{C}_{\overline{\mathbb{S}}}$ are both minimal.

In what follows, we will use a special implementation of a separator dedicated to polygons. This separator makes use of another separator which is the so-called boundary-based separator. Hence, first, we will introduce the boundary-based separator and then describe the polygon separator.

Boundary-based Separator

An alternative way to implement a separator instead of using two complementary contractors is to consider the boundary $\delta\mathbb{S}$ of the set $\mathbb{S}$. If we can provide a contractor consistent to $\delta\mathbb{S}$ and if we have a test that determines whether a given point is inside or outside $\mathbb{S}$, we can come up with a separator following the boundary approach. Let $\mathcal{C}_{\delta\mathbb{S}}$ be the contractor consistent to $\delta\mathbb{S}$ and $\mathcal{T}_{\mathbb{S}}$ a test such that

$$\mathcal{T}_{\mathbb{S}}([\mathbf{x}]) = \begin{cases} \text{true} & \text{if } [\mathbf{x}] \subseteq \mathbb{S} \\ \text{false} & \text{otherwise} \end{cases} \tag{2.55}$$

that determines if $[\mathbf{x}] \in \mathbb{R}^n$ is inside or outside $\mathbb{S}$. The boundary-based separator $\mathcal{S}_{\delta\mathbb{S}}$ performs two steps. First, the separator applies $\mathcal{C}_{\delta\mathbb{S}}$ to contract an input box $[\mathbf{x}]$. That means this operation provides a smaller box $[\delta\mathbf{x}]$, that includes the subset of $\delta\mathbb{S}$ ignoring if a subset is inside or outside $\mathbb{S}$. The resulting boxes are visualized in Figure 2.15. If we obtain $[\delta\mathbf{x}] \subseteq [\mathbf{x}]$, we can divide the initial input box $[\mathbf{x}]$ into subboxes as visualized in Figure 2.15. The number of subboxes depends on how $\delta\mathbb{S}$ and $[\mathbf{x}]$ are related to each other. In general, together with

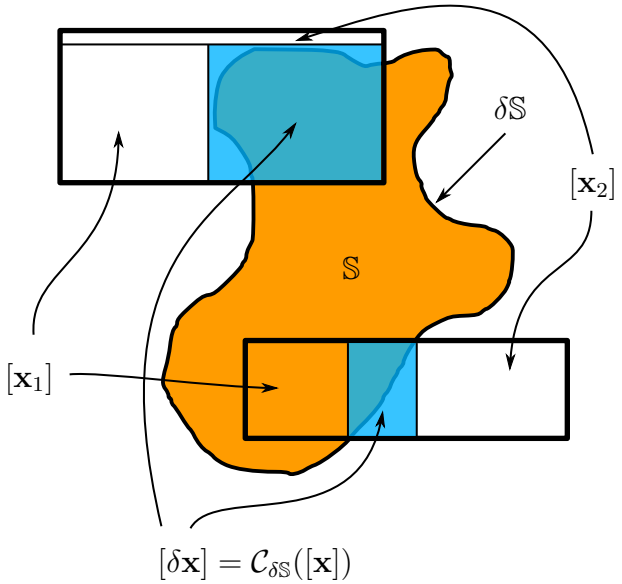


Figure 2.15: Illustration of the boundary separator based on the example introduced in Figure 2.14a. The boundary boxes $[\delta\mathbf{x}]$ are colored blue. The boundary box divides the initial box into multiple subsets that we call $[\mathbf{x}_1]$ and $[\mathbf{x}_2]$.

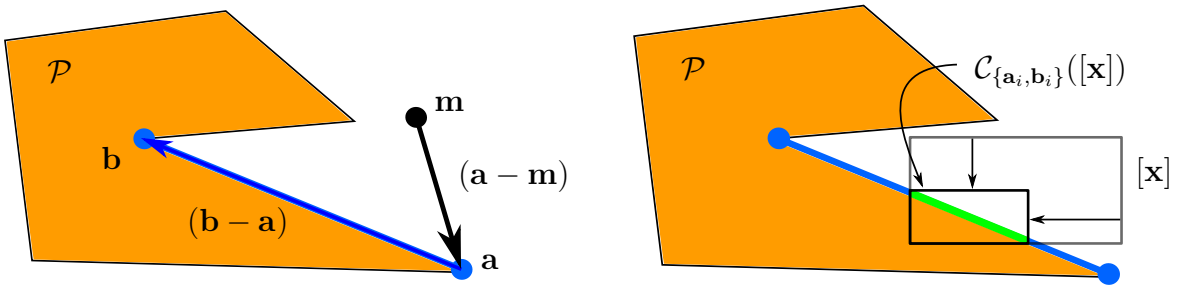
$[\delta\mathbf{x}]$, there can be at most five subboxes for the case that $[\mathbf{x}]$ fully includes $\mathbb{S}$. In Figure 2.15, the input boxes lead to three subboxes that we denote $[\mathbf{x}_1]$, $[\mathbf{x}_2]$ and $[\delta\mathbf{x}]$. Since all subboxes besides $[\delta\mathbf{x}]$ cannot cross $\delta\mathbb{S}$ – that means the remaining subboxes need to be either inside or outside $\mathbb{S}$ – we can choose in each subbox besides in $[\delta\mathbf{x}]$ an arbitrary point, for instance, the mid-point, and utilize $\mathcal{T}_{\mathbb{S}}$ to determine if the subbox is inside or outside $\mathbb{S}$. In Figure 2.15 the bottom example shows a case where $[\mathbf{x}_1]$ is inside $\mathbb{S}$ and $[\mathbf{x}_2]$ outside $\mathbb{S}$. However, special constellations such as the upper example may occur, where we obtain no interval box dedicated to the inner part since both subboxes happen to be placed outside $\mathbb{S}$. The result of the separation for the bottom example in Figure 2.15 is given by

$$\begin{aligned} \mathcal{S}_{\delta\mathbb{S}}([\mathbf{x}]) &= \{[\mathbf{x}_{\text{in}}], [\mathbf{x}_{\text{out}}]\} \\ &= \{[\delta\mathbf{x}] \cup [\mathbf{x}_1], [\delta\mathbf{x}] \cup [\mathbf{x}_2]\} \end{aligned} . \tag{2.56}$$

But for the upper example $[\mathbf{x}_{\text{in}}]$ is empty and $[\mathbf{x}_{\text{out}}]$ is determined by $[\delta\mathbf{x}] \cup [\mathbf{x}_1] \cup [\mathbf{x}_2]$. Hence, the union over all subboxes that are determined to be outside $\mathbb{S}$ and the boundary box $[\delta\mathbf{x}]$ determines $[\mathbf{x}_{\text{out}}]$. The union over all subboxes that are determined to be inside $\mathbb{S}$ and the boundary box $[\delta\mathbf{x}]$ determines $[\mathbf{x}_{\text{in}}]$.

Polygon Separator

Closed polygons well represent structured environments such as buildings and rooms. Our next step is to adapt the boundary-based contractor to polygons. Therefore let us consider the border of a polygon. Since a polygon is defined as a composition of oriented line segments, the polygon separator mirrors this configuration by multiple boundary-based contractors, each corresponding to a different segment. Therefore, let us first consider the boundary-based contractor dedicated to oriented line segments.



(a) Related variables to check if $\mathbf{m}$ lies on a line segment.

(b) Illustration of the line segment contractor $\mathcal{C}_{\{\mathbf{a}_i,\mathbf{b}_i\}}$.

Figure 2.16: The set described by a polygon is $\mathcal{P}$. The connected vertices $\mathbf{a}$ and $\mathbf{b}$ span a line segment which is a part of the boundary $\delta\mathcal{P}$. Figure 2.16a illustrates the related variables involved in the constraints in (2.57). The point $\mathbf{m}$ does not fulfill the first constraint since the vectors $\mathbf{b}-\mathbf{a}$ and $\mathbf{a}-\mathbf{m}$ are not collinear. Figure 2.16b illustrated the line segment contractor $\mathcal{C}_{\{\mathbf{a}_i,\mathbf{b}_i\}}$.

Let $\{\mathbf{a},\mathbf{b}\}$ be the line segment defined by $\mathbf{a},\mathbf{b}\in\mathbb{R}^2$ as end and start point. A point $\mathbf{m}$ lies on the segment $\mathbf{m}\in\{\mathbf{a},\mathbf{b}\}$, if the two constraints

$$\begin{cases}\det(\mathbf{b}-\mathbf{a},\mathbf{a}-\mathbf{m})=0,\\ \min(\mathbf{a},\mathbf{b})\leq\mathbf{m}\leq\max(\mathbf{a},\mathbf{b})\end{cases} \tag{2.57}$$

are fulfilled. While the first constraint determines $\mathbf{m}$ to be on the line $\mathbf{a}$ and $\mathbf{b}$ span, the second constraint checks whether $\mathbf{m}$ is within the segment determined by the end and start point. We illustrate this in Figure 2.16a. In this example, vectors $\mathbf{b}-\mathbf{a}$ and $\mathbf{a}-\mathbf{m}$ are not collinear, due to which $\mathbf{m}$ is not on the line and the first constraint is not fulfilled.

A general polygon $\mathcal{P}$ comprises n oriented line segments. We describe the i-th segment by $\{\mathbf{a}_i,\mathbf{b}_i\}$ for $i\in\{1,...,n\}$. As a consequence, the border $\delta\mathcal{P}$ of a polygon is given by

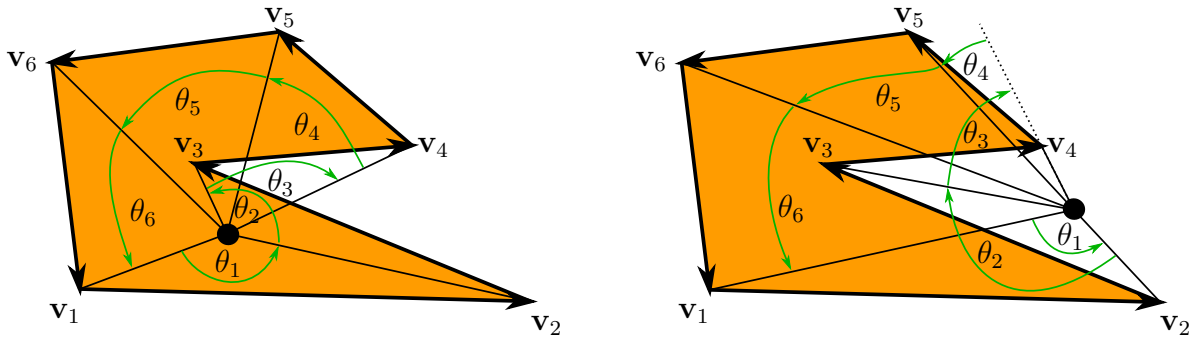
$$\delta\mathcal{P}=\{\mathbf{m}\in\mathbb{R}^2\,|\,\exists i\in\{1,...,n\},\mathbf{m}\in\{\mathbf{a}_i,\mathbf{b}_i\}\}. \tag{2.58}$$

For each of the line segments $\{\mathbf{a}_i,\mathbf{b}_i\}$ we define a contractor $\mathcal{C}_{\{\mathbf{a}_i,\mathbf{b}_i\}}$, that contracts an input box to the boundary defined by the line segment. Therefore, we use the constraints defined in (2.57) and build the forward-backward contractor described above. The line segment contractor is illustrated in Figure 2.16b.

The union of all line segment contractors defines the contractor for the whole polygon boundary:

$$\mathcal{C}_{\delta\mathcal{P}}=\bigcup_{i=1}^{n}\mathcal{C}_{\{\mathbf{a}_i,\mathbf{b}_i\}}. \tag{2.59}$$

The only component that remains to be added to complete the polygon separator is the test $\mathcal{T}_{\mathcal{P}}$, which decides if a given point lies inside or outside the polygon. To accomplish this task, [54] proposes to utilize the winding number test. This test is portrayed in Figure 2.17 and aims to evaluate the total number of times the curve consisting of line segments travels around a given point. The winding number depends on the orientation of the line segments, and therefore we need to ensure that all segments of the polygon are consistently oriented. Considering the



(a) The test point is inside since the sum of all angles amounts to 2π.

(b) The test point is outside since the sum of all angles is less than 2π.

Figure 2.17: Winding number test to validate if a point $\mathbf{m}$ is inside a polygon $\mathcal{P}$ or outside.

polygon $\mathcal{P}$ defined by an ordered set of corner points $\mathbf{v}_i \in V_\mathcal{P}$, the winding number for a given point $\mathbf{m}$ is defined by

$$wn(\mathbf{m}, \mathcal{P}) = \frac{1}{2\pi} \sum_{i=1}^{n} \theta_i \tag{2.60}$$

$$= \frac{1}{2\pi} \sum_{i=1}^{n} \arctan2 \left(\det((\mathbf{v}_i - \mathbf{m}), (\mathbf{v}_{i+1} - \mathbf{m})), (\mathbf{v}_i - \mathbf{m})^T (\mathbf{v}_{i+1} - \mathbf{m}) \right). \tag{2.61}$$

If $\mathbf{m}$ is placed outside $\mathcal{P}$, the winding number will be $wn(\mathbf{m}, \mathcal{P}) < 1$ (cf. Figure 2.17b), otherwise if $\mathbf{m}$ is inside $\mathcal{P}$ (cf. Figure 2.17a). Note that θ_i is the oriented angle from $(\mathbf{v}_i - \mathbf{m})$ to $(\mathbf{v}_{i+1} - \mathbf{m})$. Now we have assembled all the necessary components to build the boundary-based separator $\mathcal{S}_\mathcal{P}$ for the polygon $\mathcal{P}$ as described above. We first apply the polygon-boundary contractor $\mathcal{C}_{\delta\mathcal{P}}$ to obtain the subboxes as illustrated in Figure 2.15 and after that assign the subboxes to the inner and outer parts by applying the in-polygon test $\mathcal{T}_\mathcal{P}$ utilizing the winding number test for an arbitrary point inside each subbox.

2.2.3 Pessimism and Wrapping Effect

One of the main challenges of applying interval analysis to robotics is pessimism, meaning an overestimation of the uncertainty and, therefore, may end up with meaningless results. The pessimism is mainly caused by two problems we want to discuss briefly in this section. However, although we tend to overestimate the uncertainty, the reliability of the result is not affected. Nonetheless, dealing with interval analysis, it is crucial to consider overcoming pessimism.

2.2.3.1 Dependencies between variables

Let us consider the following example.

Example 2.2.7. We subtract the same intervals:
$[1,2] - [1,2] = [\{a - b \,|\, a \in [1,2], b \in [1,2]\}] = [1-2, 2-1] = [-1, 1]$.

The subtraction is performed strictly according to the inclusion function of the subtraction operator. We can generalize the example to the difference between two identical non-degenerate intervals $[x] - [x] = [\underline{x} - \overline{x}, \overline{x} - \underline{x}]$ which is not thin. While subtracting the same element in

$\mathbb{R}$ leads to the neutral element of addition 0, the interval counterpart in $\mathbb{IR}$ does not naturally lead to a thin result. This example again highlights that the properties of basic operations on intervals can differ from their equivalent in $\mathbb{R}$. This issue appears if a variable is involved several times in an expression, as in the example. The dependency between the variables directly leads to unwanted pessimism.

A modification of the analytical expression is vital to avoid or at least reduce this effect. We can reduce the pessimism implied by the dependency between variables by exploiting and deleting common subexpressions to obtain a minimal expression representation. However, up to now, there are no general methods adopted in existing interval solvers.

2.2.3.2 Wrapping effect

As illustrated in Figure 2.10, we introduce pessimism whenever we try to enclose a set that is not an axis-aligned box since intervals and interval boxes are axis-aligned elements. We call this phenomenon the wrapping effect. The successive evaluation of a function that suffers from over-enclosure can cause pessimism to overgrow. To illustrate this problem, Moore introduces in his book [33] an example where he applies consecutive rotations to a box. We give a similar example in Figure 2.18. To overcome the wrapping effect, dividing the solution space into a set of non-overlapping boxes (subpaving) and individually contracting those subsets is helpful. While this approach significantly reduces pessimism, it comes with higher computation and memory consumption costs. The following section provides more insights into that approach.

2.2.4 Set Inversion Via Interval Analysis (SIVIA)

An arbitrary solution set that may contain holes cannot be approximated by an interval vector without introducing pessimism due to the wrapping effect. One way to cope with that problem is to divide the solution space into multiple non-overlapping boxes and to evaluate each box to see if it contains parts that belong to the solution set. As a result, instead of a box wrapped around the solution set, the goal, for now, is to represent the approximation of the solution set by a set of boxes – called subpaving. In particular, subpavings become handy for set-inversion problems.

Why shall we consider the inversion of a function in the context of robotics? To answer this question, let us consider, for instance, the localization problem. Let $\mathbf{x}$ be the unknown pose of the robot. Further, let $\mathbf{y}$ be the measurements the robot perceives from its pose $\mathbf{x}$ in the mapped environment $\mathbf{m}$. Typically, in robotics, the measurement function $\mathbf{f}(\mathbf{x}) = \mathbf{y}$ that maps a pose to local measurements under consideration of the map is known. This function is an analytical description of how the utilized sensor perceives its environment $\mathbf{m}$. However, in the localization problem, the local measurements $\mathbf{y}$ are known, and the goal is to determine the pose $\mathbf{x}$ given the local measurements $\mathbf{y}$ and the map $\mathbf{m}$. That means, to solve the localization problem, we need to invert the measurement function $\mathbf{f}^{-1}(\mathbf{y}) = \mathbf{x}$ given the map $\mathbf{m}$ to obtain the pose. SIVIA is an inversion approach for sets that uses subpavings. In the following, we will introduce subpavings and the SIVIA algorithm.

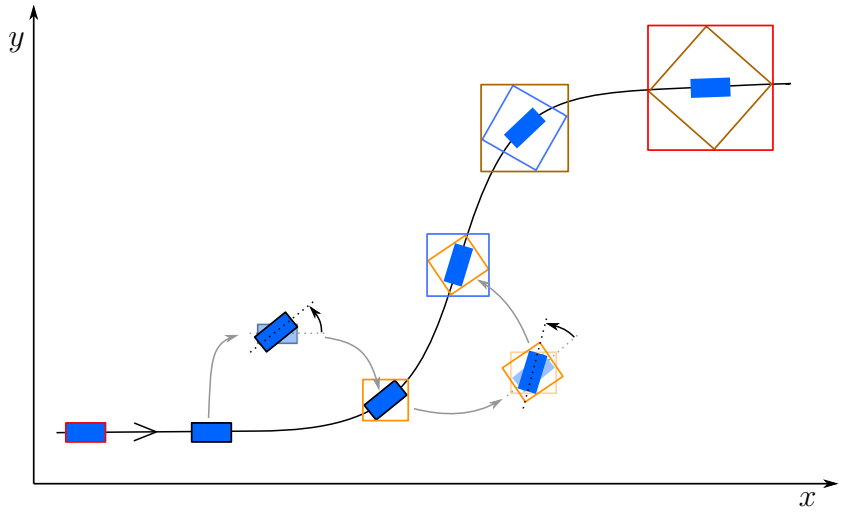


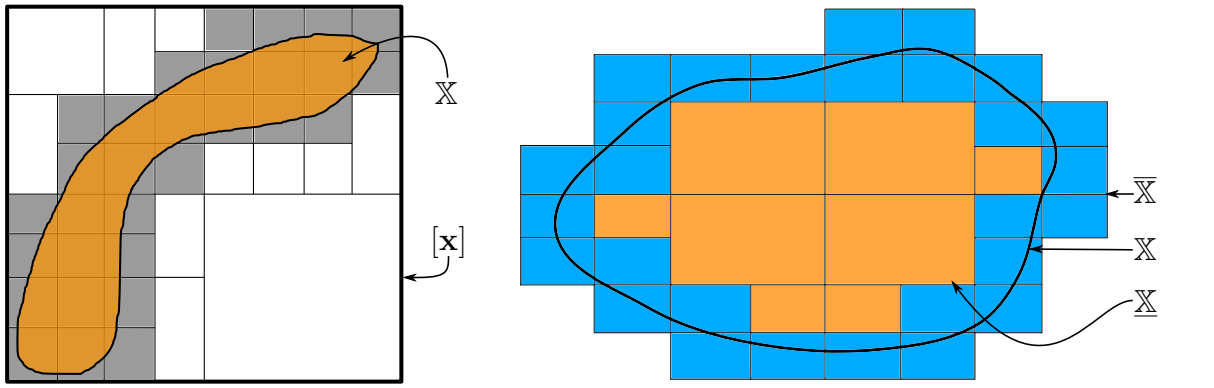
Figure 2.18: Illustration of the wrapping effect when a robot performs rotation. A black line visualizes the robot's trajectory, and the positions at different points in time are illustrated with blue boxes. The robot starts the trajectory in the bottom left. Here, the pose uncertainty is very small, shown by a red interval box. Also, for the next pose in the trajectory, the enclosing interval box (black) is small since no rotation was involved. However, the robot rotates between the second and third pose, as illustrated in the figure. The axis-aligned box that encloses the rotated black box is colored orange. Here, the wrapping effect leads the orange box to overestimate the robot's position due to the property of an axis-aligned box. The movement from the third to the fourth position again involves a rotation, and further overestimation is introduced. The successive rotation of the robot leads to a large enclosure (red box at the end) that highly overestimates the position uncertainty.

2.2.4.1 Subpaving

The approximation of an arbitrary set $\mathbb{X} \subset \mathbb{R}^n$ with a union of non-overlapping boxes

$$\mathbb{K} = \{[\mathbf{x}_1], ..., [\mathbf{x}_n]\} \tag{2.62}$$

is usually thinner compared to a single box $[\mathbf{x}]$ that contains $\mathbb{X}$ due to the wrapping effect. We call the set of all boxes $\mathbb{K}$ a subpaving. We can obtain a subpaving by applying a finite number of bisections and selections on $[\mathbf{x}]$ as illustrated in Figure 2.19a. The initial box $[\mathbf{x}]$ is the large box that fully contains the set $\mathbb{X}$. By recursively bisecting $[\mathbf{x}]$ and selecting those parts that contain the solution set, we obtain a set of subboxes as a union that approximates $\mathbb{X}$ with less pessimism compared to $[\mathbf{x}]$. We call this subpaving regular. For interval boxes in higher dimensions, the question of how to bisect those boxes may arise. Typically the widest dimension is chosen to be bisected. This is also why the subboxes in Figure 2.19 have different sizes. However, selecting the widest dimension is just a heuristic that has proven to provide good results in computing a subpaving efficiently. That means another strategy to divide the solution space is also valid and sometimes more favorable depending on the solution set we seek to approximate.



(a) Illustration of a subpaving as a result of recursive bisection and selection.

(b) Outer and inner subpaving. While the inner subpaving $\underline{\mathbb{X}}$ is colored orange, the outer subpaving $\overline{\mathbb{X}}$ blue.

Figure 2.19: Visualization of regular subpavings.

Generally, two subpavings are used to approximate a set $\mathbb{X}$. While the inner subpaving $\underline{\mathbb{X}}$ only consists of boxes that are fully included inside $\mathbb{X}$, the outer subpaving defines the outer approximation and is denoted by $\overline{\mathbb{X}}$. For example, the subpaving in Figure 2.19a is an outer subpaving. As a consequence,

$$\underline{\mathbb{X}} \subseteq \mathbb{X} \subseteq \overline{\mathbb{X}} \tag{2.63}$$

holds. Figure 2.19b visualizes the different types of subpavings. Usually, we are interested in the outer approximation of a set, and therefore, in this work's scope, we will focus on the outer subpaving.

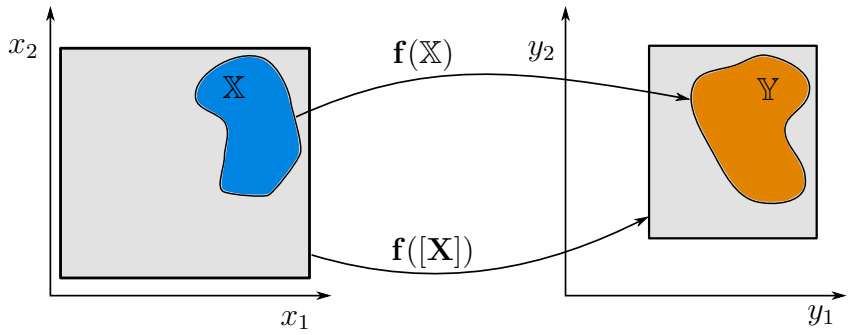
2.2.4.2 SIVIA algorithm

The SIVIA algorithm is a set inversion approach that allows approximating the set $\mathbb{X}$ for an arbitrary image set $\mathbb{Y}$ that are linked by a possibly non-linear function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ by

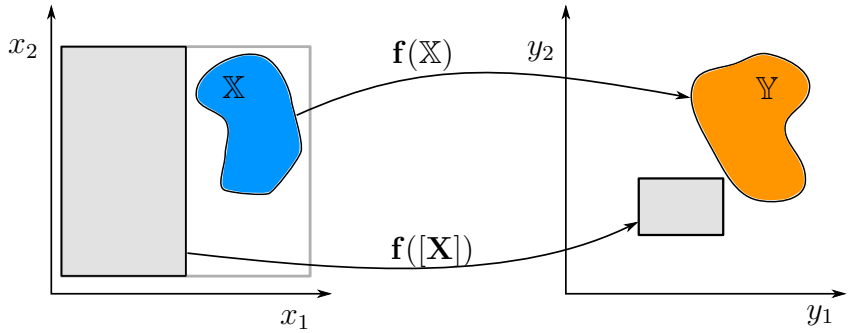
$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} = \mathbf{f}^{-1}(\mathbb{Y}). \tag{2.64}$$

SIVIA is a branch and bound algorithm that utilizes the subdivision of the solution space to compute the inner and outer subpaving enclosing the solution set. Therefore, it uses the inclusion function $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^m$. An exemplary version of the SIVIA algorithm is shown in Algorithm 2. The algorithm needs as an input the inclusion function $[\mathbf{f}]$ and the image set $\mathbb{Y}$ as described above. Furthermore, we need to provide an initial (possibly very large) box $[\mathbf{x}_0]$ that encloses the outer approximation $\overline{\mathbb{X}}$ of the solution set $\mathbb{X}$. The accuracy of the outer approximation depends on an additional parameter ϵ that determines the minimal width of a box that we consider for bisection. Internally, SIVIA uses a stack that stores all those boxes that need to be evaluated. Hence, the initial large box $[\mathbf{x}_0]$ is inserted into the stack as depicted in line 1. We can encounter four cases for each box stored in the stack. For reader convenience, we also provide a visual explanation of the successive steps of the SIVIA algorithm and the different possible cases in Figure 2.20.

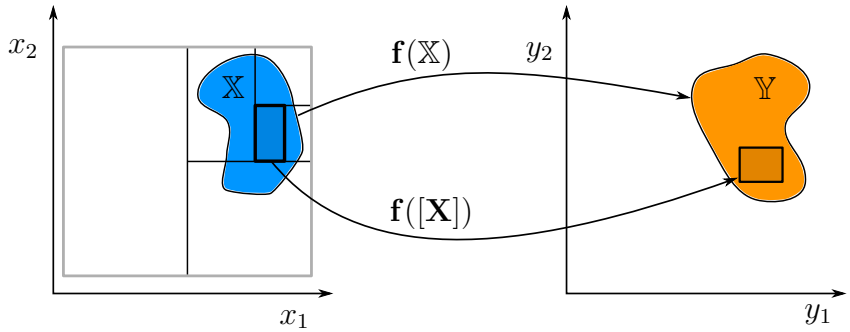
1. If $[\mathbf{f}]([\mathbf{x}])$ has an empty intersection with $\mathbb{Y}$, then $[\mathbf{x}]$ does not belong to $\mathbb{X}$ and does not need to be further considered as shown in Algorithm 2 lines 4 and 5 (cf. Figure 2.20b).



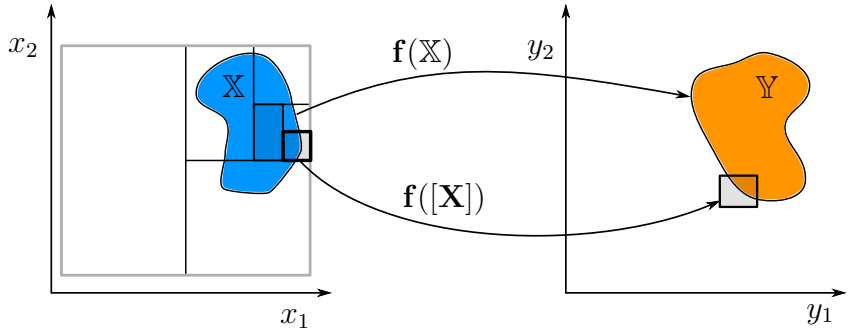
(a) $[\mathbf{f}]([\mathbf{x}])$ has a non-empty intersection with $\mathbb{Y}$ and the width is larger than ϵ. Hence, $[\mathbf{x}]$ needs to be bisected.



(b) $[\mathbf{f}]([\mathbf{x}])$ has no intersection with $\mathbb{Y}$. As a result, $[\mathbf{x}]$ cannot be part of the solution $\mathbb{X}$ and can be discarded.



(c) $[\mathbf{f}]([\mathbf{x}])$ is fully contained in $\mathbb{Y}$. The box $[\mathbf{x}]$ is part of the inner subpaving $\underline{\mathbb{X}}$ as it does not contain any parts that are not part of the solution set.



(d) $[\mathbf{f}]([\mathbf{x}])$ has a non-empty intersection with $\mathbb{Y}$ but the width of $[w(x)]$ is smaller than ϵ. That means $[w(x)]$ is not further bisected and is considered as part of the outer subpaving $\overline{\mathbb{X}}$.

Figure 2.20: Visual explanation of the successive steps of the SIVIA algorithm with the different possible cases.

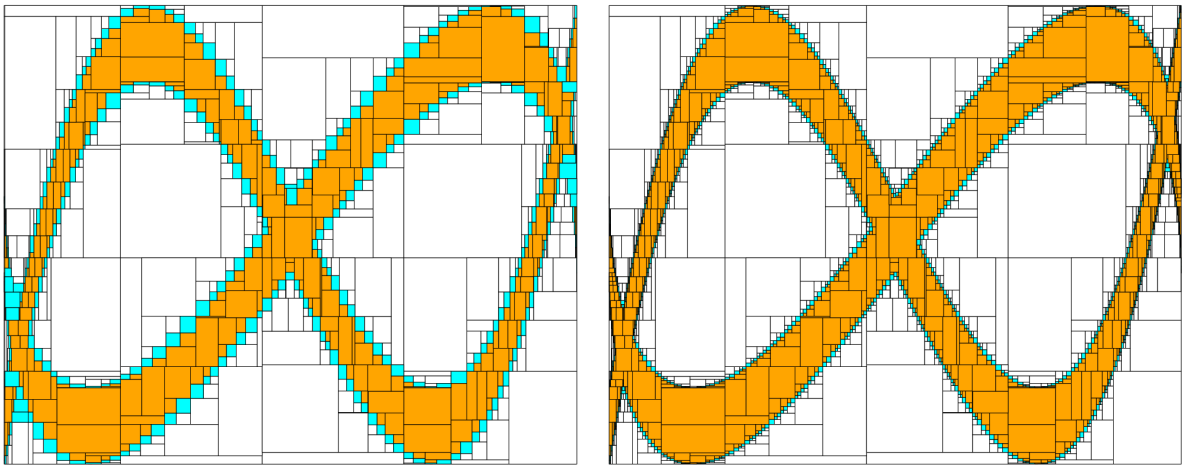
Algorithm 2: SIVIA

```
   Data: [f], [x_0], 𝕐, ϵ
   Result: 𝕏̲, 𝕏̄
   // Initialize the stack S.
1  S.push([x_0]);
2  while S ≠ ∅ do
3      [x] =top(S);
4      if [f]([x]) ∩ 𝕐 = ∅ then
           // [x] cannot be part of the solution set.
5          continue;
6      else if [f]([x]) ⊂ 𝕐 then
           // [x] is an inner subset of 𝕏.
7          𝕏̲ = 𝕏̲ ∪ [x];
8          𝕏̄ = 𝕏̄ ∪ [x];
9      else if w([x]) < ϵ then
           // [x] partially overlaps with 𝕏 but too small for bisection.
10         𝕏̄ = 𝕏̄ ∪ [x];
11     else
           // [x] partially overlaps with 𝕏, needs to be bisected.
12         ([x_1], [x_2]) = bisect([x]);
13         S.push([x_1]);
14         S.push([x_2]);
15     end
16 end
```

2. If $[\mathbf{f}]([\mathbf{x}])$ is entirely in $\mathbb{Y}$, then $[\mathbf{x}]$ belongs to the solution set $\mathbb{X}$ and needs to be inserted to $\overline{\mathbb{X}}$ and $\underline{\mathbb{X}}$ as presented in Algorithm 2 lines 6 to 8 (cf. Figure 2.20c).
3. If $[\mathbf{f}]([\mathbf{x}])$ partially overlaps but is not entirely in $\mathbb{Y}$ and if the width of the box is lower than ϵ, then it is not bisected and is determined to be only part of the outer approximation $\overline{\mathbb{X}}$ as described in Algorithm 2 lines 9 and 10 (cf. Figure 2.20d).
4. Otherwise, we bisect $[\mathbf{x}]$ and push the resulting subsets to the stack to be considered in the successive iterations (Algorithm 2 lines 11 to 13 and cf. Figure 2.20a).

The parameter ϵ determines the precision of the approximation of the solution set. The smaller ϵ, the more bisections are performed, and the smaller the boxes on the outer region of the outer subpaving will get. Nonetheless, this comes with the cost of higher computational effort and memory consumption. Figure 2.21 shows examples of different accuracies for the SIVIA-based computation of a set.

The branch and bound approach that SIVIA introduces is not only handy for set inversion problems. Also, in the case of pessimistic contractors, SIVIA can increase the accuracy of the result by applying the contractor on subsets of the initial box. Algorithm 3 shows a simple version of the SIVIA algorithm, which utilizes a contractor $\mathcal{C}$. Notice that the contractor $\mathcal{C}$ is used to reduce the size of the box $[\mathbf{x}]$ in line 4 before it is bisected. As a result, coupling the SIVIA algorithm with contractors decreases the time complexity since potentially fewer bisections become necessary for a fixed accuracy ϵ. Although applying a contractor in the SIVIA algorithm provides more accurate results, the computational effort and memory consumption increase compared to the contractor-only approach. Unfortunately, the computation time



(a) Low accuracy since ϵ is large. (b) High accuracy since ϵ is small.

Figure 2.21: Representation of a set by subpavings computed by SIVIA with different accuracy levels. The outer subpaving is colored blue and the inner subpaving is colored orange.

Algorithm 3: SIVIA with contractor

Data: $\mathcal{C}$, $[\mathbf{x}_0]$, ϵ
Result: $\overline{\mathbb{X}}$
// Initialize the stack S.
S.push($[\mathbf{x}_0]$);
while $S \neq \varnothing$ **do**
  $[\mathbf{x}] =$top(S);
  $[\mathbf{x}] = \mathcal{C}([\mathbf{x}])$;
  if $[\mathbf{x}] = \varnothing$ **then**
    // $[\mathbf{x}]$ cannot be part of the solution set.
    continue;
  else if $w(\mathbf{x}) < \epsilon$ **then**
    // $[\mathbf{x}]$ partially overlaps with $\mathbb{X}$ but too small for bisection.
    $\overline{\mathbb{X}} = \overline{\mathbb{X}} \cup [\mathbf{x}]$;
  else
    // $[\mathbf{x}]$ partially overlaps with $\mathbb{X}$, needs to be bisected.
    $([\mathbf{x}_1], [\mathbf{x}_2]) =$ bisect($[\mathbf{x}]$);
    S.push($[\mathbf{x}_1]$);
    S.push($[\mathbf{x}_2]$);
  end
end

increases exponentially with the dimension of $\mathbf{x}$. If we have fewer dimensions, utilizing SIVIA to improve the results is a good choice. However, the more dimensions we consider for bisection, the more subsets must be evaluated. So, in the case of more dimensions, it is worth prioritizing the dimensions and only bisecting those (hopefully) few dimensions mainly affected by the pessimism of the contractor.

2.2.5 Relaxed intersection

The core assumption for the application of interval analysis is that the correct value $\mathbf{x}^*$ is contained in the interval $[\mathbf{x}]$. That means if we have n measurements that we use to construct n measurement intervals $[\mathbf{x}_i]$ with $i \in 1, ..., n$ taking the measurement uncertainty into account, we assume that all measurement intervals enclose the true value $\mathbf{x}^*$ that we would obtain if we would have a perfect error-free measurement. The selection of the error bounds of $[\mathbf{x}_i]$ need to be carefully chosen. If we account for large errors that might happen on rare occasions, the interval widths will inflate significantly, and we will obtain large meaningless intervals. On the other hand, if we choose the bounds too tight, the assumption that $\mathbf{x}^*$ has to be contained in $[\mathbf{x}_i]$ can be violated. That means choosing the interval bounds as tight as possible and as wide as necessary is vital for interval-based approaches.

Since gross errors like outliers are rare, considering those large errors in selecting the error bounds often leads to large meaningless intervals. Furthermore, in many cases, it is not even possible to provide error bounds on outliers. That means we need to take such gross errors differently into account. Therefore, we introduce the notion of relaxed intersection in this section.

Let us consider a simple example.

Example 2.2.8. Let the four intervals $[x_1] = [-1, 1]$, $[x_2] = [-2, 0]$, $[x_3] = [-1, 3]$ and $[x_4] = [1, 2]$ be the measurements of same value. However, the intersection of all intervals is empty

$$[x_1] \cap [x_2] \cap [x_3] \cap [x_4] = \varnothing. \tag{2.65}$$

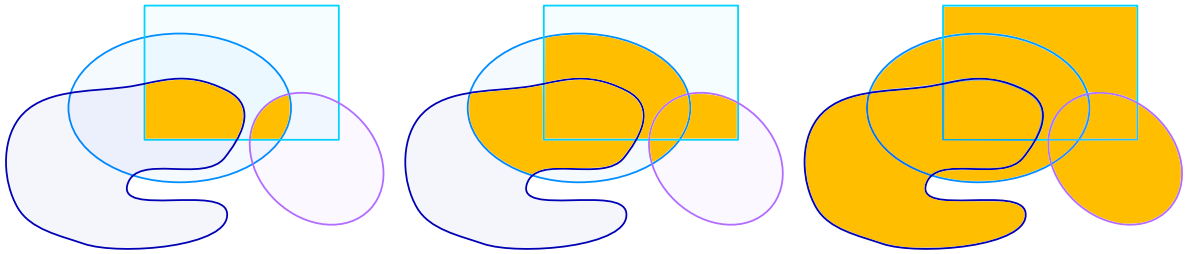
To allow outliers in our interval-based computations, it is possible to perform a *q-relaxed* intersection [55]. This robust intersection approach tolerates maximum q outliers while all other intervals have to overlap. Therefore, we need to specify the number q that reveals how many outliers we encounter. We denote the q-relaxed intersection by $\bigcap_{1 \leq i \leq n}^{\{q\}} [\mathbf{x}_i]$.

Figure 2.22 visually illustrates the relaxed intersection for different values for q. Considering Example 2.2.8, the 1-relaxed intersection results into

$$\bigcap_{1 \leq i \leq 4}^{\{1\}} [\mathbf{x}_i] = [-1, 0], \tag{2.66}$$

while the 2-relaxed intersection is

$$\bigcap_{1 \leq i \leq 4}^{\{1\}} [\mathbf{x}_i] = [-1, 2]. \tag{2.67}$$



(a) $\bigcap_{1\leq i\leq 4}^{\{1\}} [\mathbf{x}_i]$. (b) $\bigcap_{1\leq i\leq 4}^{\{2\}} [\mathbf{x}_i]$. (c) $\bigcap_{1\leq i\leq 4}^{\{3\}} [\mathbf{x}_i]$.

Figure 2.22: Illustration of the q-relaxed intersection.

2.3 Sensor Models

In this section, we introduce the sensors we employed in this work. The introduction includes a general overview of the operating principle and a model of each sensor. Since this work focuses on robot localization, our sensors can generally be categorized into global and local. While global sensors provide information about the global location of the vehicle, local sensors provide information about the vehicle's immediate environment. While the cameras and LiDAR sensors provide local data, GNSS sensors provide global positioning data. First, we will introduce stereo cameras and LiDAR sensors in Subsection 2.3.1 and 2.3.2. Finally, the global GNSS sensor is introduced in Subsection 2.3.3.

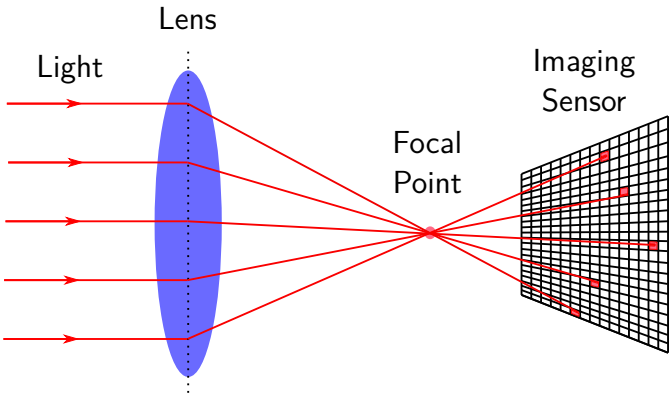
2.3.1 Stereo Cameras

In this work, we use a stereo camera system that combines rich projective image information and provides depth data. First, we introduce the monocular pinhole camera model in Part 2.3.1.1. Based on the monocular camera model, we will consider the multi-view stereo vision case and provide a summary of the epipolar geometry in Part 2.3.1.2, which is the basis for interval-based feature reconstruction.

2.3.1.1 Monocular Camera Model

A figure of a camera that we used in our experiments and a schematic structure are shown in Figure 2.23. In contrast to LiDAR sensors, optical cameras are typically passive measurement systems. An optical lens system first focuses visible light passing through the camera aperture. Figure 2.23b illustrates the lens system by a simplified single lens. The bundled light is refracted to a focal point, called the projection or optical center. The actual sensor is installed behind the projection center. The imaging sensor consists of several tiles, referred to as pixels. The tiles are coated with semi-conductive materials. When visible light hits the tiles, a voltage difference is generated due to the photoelectric effect, which is measured electronically. The higher the intensity of the light, the higher the generated voltage. The most commonly used imaging sensors are Charged-Coupled Devices (CCD) and Complementary Metal-Oxide Semiconductor

(a) FLIR camera with a lens system.



(b) Simplified illustration of the operation of a projective camera.

Figure 2.23: Figure 2.23a shows a FLIR Grasshopper3 camera equipped with a Fujinon lens. Figure 2.23b depicts a simplified schematic of a camera's operation.

(CMOS) sensors. As a consequence, the imaging process performs a discretization of the continuous environment to a discretized image plane. The image resolution is determined by the number of pixels on the imaging sensor.

To mathematically describe the mapping between the 3D environment and the 2D image, the pinhole camera model is used [56]. Although in the actual camera system, as shown in Figure 2.23, the focal point is generated by the lens system, the pinhole model approximates the projection by idealizing the camera aperture by a point that coincides with the projection center and defines the origin of the optical camera frame C. Note that the approximation neglects the lens's distortion behavior, and that is why the pinhole model needs a distortion correction which we will introduce later. Figure 2.24 shows the pinhole model. To simplify the schematic figure, the image plane (depicted in gray) is projected in front of the pinhole. In reality, however, it is placed behind the projection center resulting in an image that is rotated by $180°$. Thus, we exploit the point-symmetry of the camera projection. Based on the pinhole model, the projection of a 3D point onto the 2D image plane is defined by

$$\lambda \begin{pmatrix} {}^{I}\mathbf{p} \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} {}^{I}p_x \\ {}^{I}p_y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{C}p_x \\ {}^{C}p_y \\ {}^{C}p_z \end{pmatrix} = \mathbf{K} \cdot {}^{C}\mathbf{p}. \tag{2.68}$$

The 3D point ${}^{C}\mathbf{p} = \begin{pmatrix} {}^{C}p_x & {}^{C}p_y & {}^{C}p_z \end{pmatrix}^T$ is described in the camera frame that is shown in Figure 2.24. The image pixel point is described by ${}^{I}\mathbf{p} = \begin{pmatrix} {}^{I}p_x & {}^{I}p_y \end{pmatrix}^T$ and defines the pixel position in the image frame I. The focal lengths f_x, f_y and the principal point $\begin{pmatrix} c_x & c_y \end{pmatrix}^T$ are intrinsic camera parameters that define the projection matrix $\mathbf{K}$. The intrinsic camera parameters are determined by a calibration procedure beforehand, and we assume to know those parameters in the scope of this work. Note that the projection of a 3D point to the 2D image plane is invariant to scale. That means if ${}^{C}\mathbf{p}$ is located at a different distance concerning the camera center along the observation ray, the same pixel is captured. As a result, it is impossible to reconstruct a 3D point unambiguously only from its pixel point in the image.

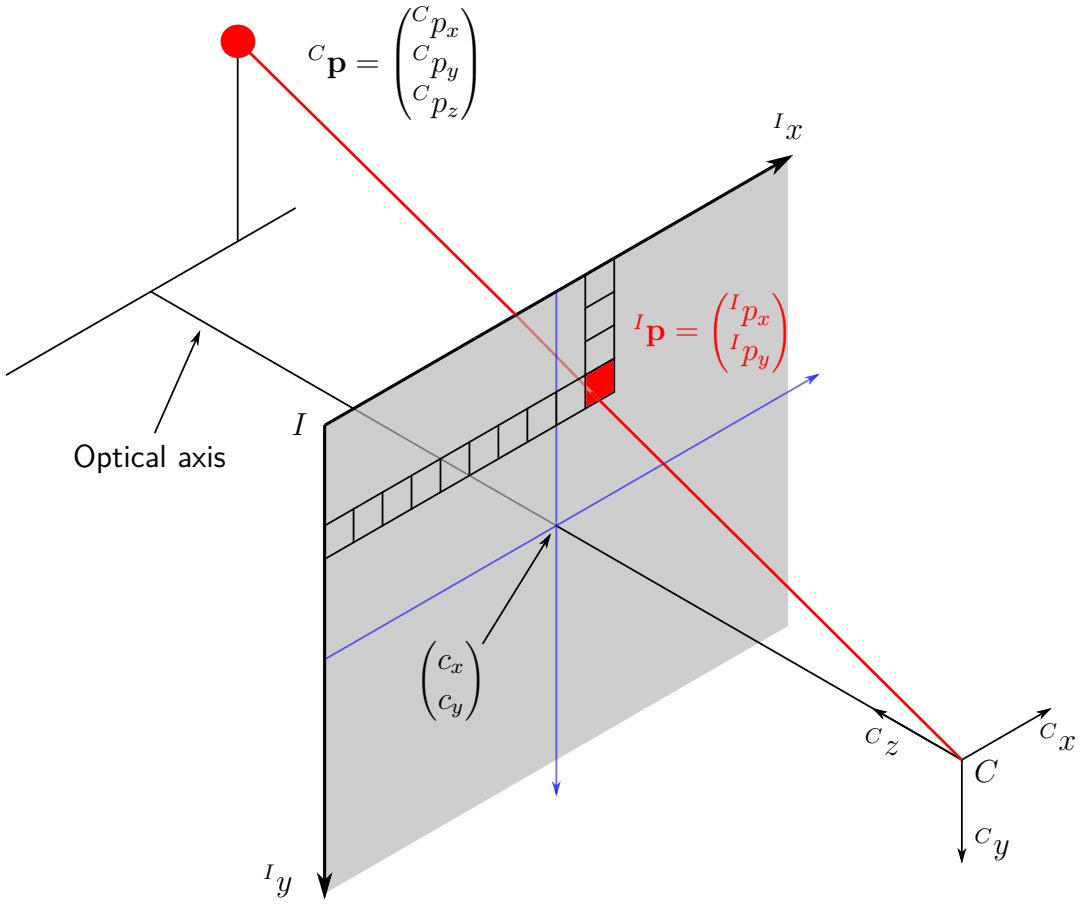


Figure 2.24: Illustration of the pinhole camera model. The figure is adopted from [51].

That is why in (2.68) on the left side, the unknown scale factor λ is introduced. Note that λ determines the distance of the measured point to the projection center. Since the camera does not capture this information, the camera only provides bearing information for points detected on the image plane.

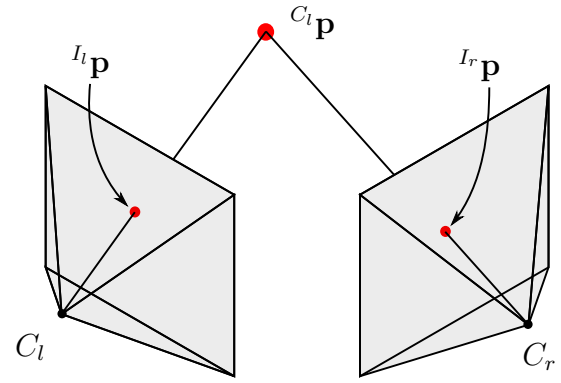
Using the pinhole model, we have neglected the distortion of the lens system up to this point. In the literature, typically, the radial and tangential distortions are considered. To compensate for the distortion, the image pixel positions must be corrected, often called undistortion or rectification. After removing the distortion, the pinhole model becomes applicable. To remove the distortion, the radial and tangential distortion parameters k_1, k_2, k_3, p_1, and p_2 need to be determined during the camera calibration. The correction of a pixel ${}^I\mathbf{p} = \begin{pmatrix} x & y \end{pmatrix}^T$ is determined by

$$r^2 = x^2 + y^2, \tag{2.69}$$

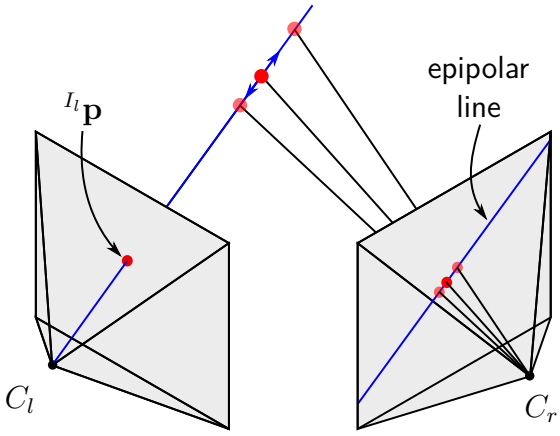
$$x' = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 xy + p_2(r^2 + 2x^2), \tag{2.70}$$

$$y' = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y^2) + 2p_2 xy. \tag{2.71}$$

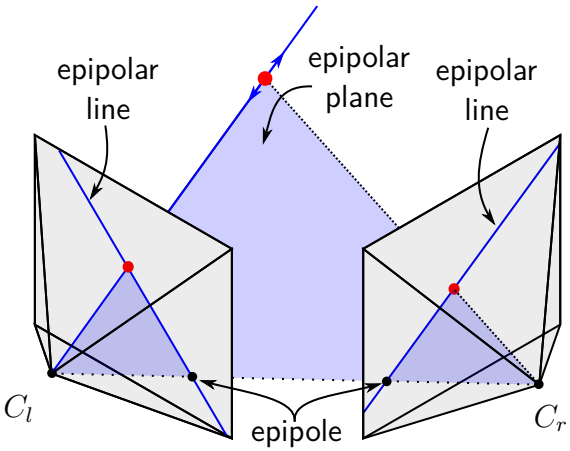
The new pixel position ${}^I\mathbf{p}' = \begin{pmatrix} x' & y' \end{pmatrix}^T$ is the distortion-free pixel position and satisfies the pinhole model. In the scope of this work, we assume that the distortion parameters are determined by the calibration accurately so that we rectify the images before further processing them.



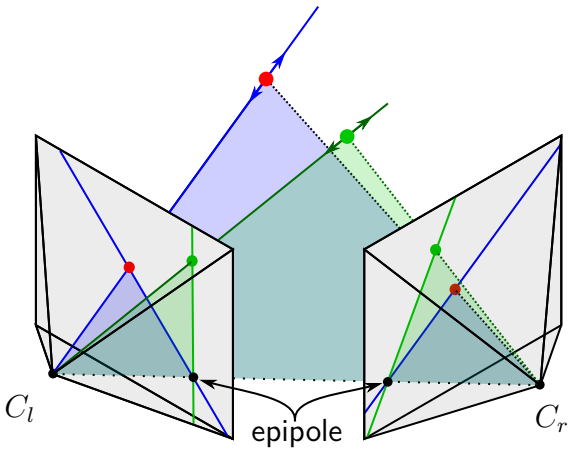
(a) Triangulation of the 3D point $^{C_l}\mathbf{p}$ when the image point correspondence is known.



(b) If the image point correspondence is unknown, the point has to lie on the epipolar line.



(c) The epipolar plane is spanned between the camera projection centers and the observed 3D point. The intersection of the epipolar plane with the image planes determines the epipolar lines.

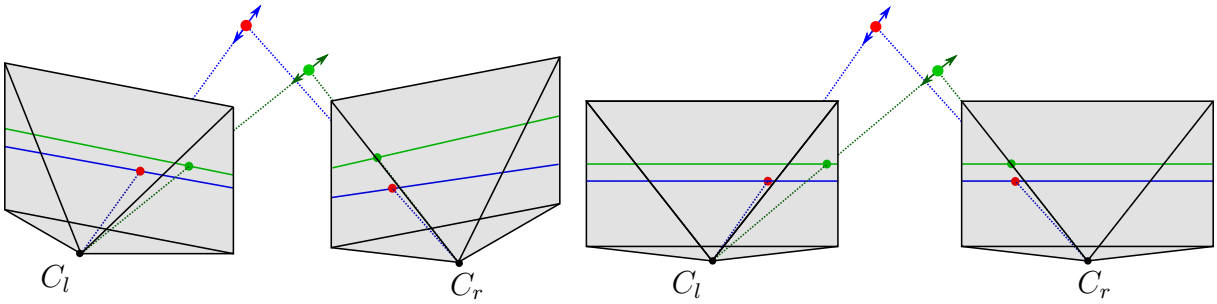


(d) Epipolar lines always intersect with the epipole. The epipole is defined by the intersection of the baseline that connects the camera projection centers with the image planes.

Figure 2.25: Epipolar geometry for two cameras. The gray cones represent cameras. The image planes are facing forward so that the FoV of both cameras overlap. The epipolar geometry describes the correspondence between the left and right image points.

2.3.1.2 Stereoscopic Model – Epipolar Geometry

A monocular camera only provides bearing information as the depth information is lost due to the projection of the scene to the image plane. To reconstruct the depth information, depth sensing technologies like LiDAR can be utilized as suggested in [57]. However, LiDAR sensors provide a relatively sparse representation of the scene compared to cameras. Another way to determine the scale is to use another camera that sees the same scene from another perspective. If the calibration parameters and the relative transformation between the cameras are known, the depth of the commonly observed 3D point can be determined from triangulation as illustrated in Figure 2.25a. However, initially, we do not know which image points of the left and right image correspond to the same point in the 3D scene. Stereo matching is a non-trivial



(a) The stereo cameras are almost parallel. (b) The cameras are perfectly parallel.

Figure 2.26: The more similar the camera's orientations are, the more parallel the epipolar lines become since the epipoles are shifted further apart. For the stereo-rectified setup, the epipoles are shifted to an infinite distance, due to which the epipolar lines degenerate into horizontal lines.

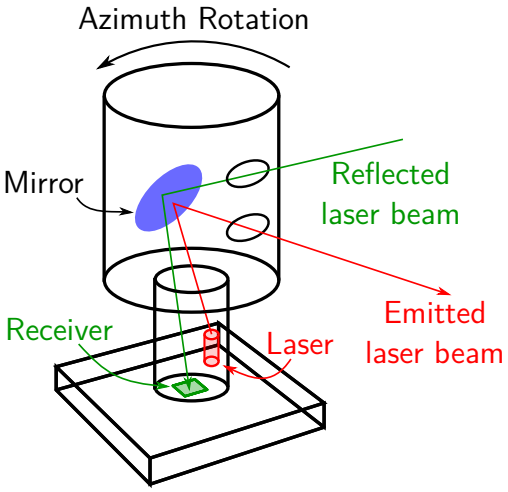
problem with a long research history in computer vision [56]. To determine if a point in the left corresponds to a point in the right image, we use feature detectors that extract distinct features in the images and determine so-called descriptors. Descriptors represent a formal description of a feature point and are unique. By comparing the descriptors of feature points in the left and right images, we can match the features with the most similar descriptors. As a result, based on the feature matching, we know which points on the left image correspond to points in the right image that captures the same 3D point in the scene.

Now that we know which image points correspond, we can continue with the geometric reconstruction of the 3D point. Therefore, let us consider again the pixel point in the left image. As the pixel point spans an observation ray on which the observed 3D point has to lie (cf. Figure 2.25b), the observation ray of potential positions generates a line of potential corresponding pixel locations in the right image. This line is the so-called epipolar line. Furthermore, the projection centers of both cameras and the 3D point span the so-called epipolar plane (blue plane in Figure 2.25c). The epipolar plane inherits the baseline between both cameras. The intersection of the epipolar plane with the image planes defines the epipolar lines. As illustrated in Figure 2.25d, all epipolar lines intersect at one point because all epipolar planes inherit the baseline. This point is the so-called epipole.

The epipolar geometry between two cameras is solely determined by the relative pose of the cameras. Figure 2.25 demonstrates the epipolar geometry for an arbitrary orientation between the cameras. Since matched image points lie on corresponding epipolar lines, obtaining a convenient orientation of the epipolar lines is favorable to accelerate the stereo feature matching. Figure 2.26 depicts the epipolar lines for different relative orientations of the cameras. The edge case where the cameras are oriented exactly parallel represents the most convenient configuration at which the epipolar lines degenerate to parallel horizontal lines in the image plane. This configuration is, in particular, convenient for stereo feature matching, as corresponding features are located on the same image row, simplifying the feature matching.

However, in practice, obtaining a perfect parallel orientation of the left and right cameras is impossible. That is why in stereoscopic vision, often a stereo-rectification is performed. The core idea of the stereo-rectification is that if we have almost parallel cameras that are slightly rotated with respect to each other, to correct the images in such a way that we obtain

(a) The Velodyne HDL-64E is an exemplary mechanical LiDAR with 64 scanlines. [58]



(b) Simplified illustration of the operation of an exemplary mechanical LiDAR based on [59].

Figure 2.27: A mechanical LiDAR has mechanically rotating parts that cover a large FoV.

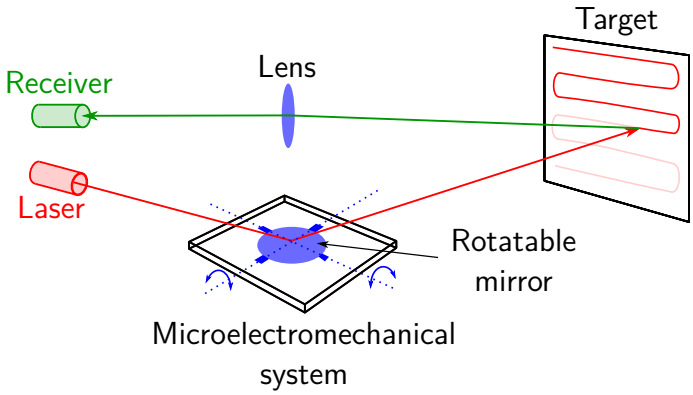
images that are identical to images that perfectly parallel cameras would have taken. The core ingredient of the stereo-rectification is to apply a so-called homography that rotates the images accordingly. This type of transformation maps points from one plane to another plane – in our case, a slightly rotated plane to obtain the stereo-rectification. The stereo camera calibration determines the homography matrix. While the monocular camera calibration determines the intrinsic camera parameters, the stereo calibration determines the relative transformation between the stereo camera pair used for the stereo-rectification. In the scope of this work, we assume that the calibration parameters are accurately known. Furthermore, before applying our algorithms to the stereo images, we stereo-rectify them to fulfill the assumption of parallel cameras.

2.3.2 Light Detection And Ranging (LiDAR) sensors

Light Detection And Ranging (LiDAR) sensors are laser scanners classified as active optical measurement systems. LiDAR sensors rely on the Time of Flight (ToF) principle. The distance between the sensor and the reflecting target is determined by measuring the time delay between the emission and reception of the light signal and by considering the speed of light. The wavelength of the emitted light depends on the field of application. However, infrared light with $905\,\mathrm{nm}$ is typically used in the automotive sector and robotics. In recent years, new LiDAR technologies have been developed. Besides the traditional pulsed mechanically rotating LiDAR, new ranging mechanisms, beam generation, and deflection technologies are becoming State of the Art. We want to briefly summarize here and focus on the LiDAR technologies utilized in the scope of this work.

LiDAR sensors are typically categorized concerning the ranging technology and the mechanical structure. The ranging technology defines how the depth information is deduced from the emitted and received signal. Therefore, different ranging technologies employ different types of light signals. The LiDAR ranging technologies can be categorized into three different

(a) The Cepton Vista 8800 is a pulsed MEMs solid state LiDAR. [60]



(b) Simplified illustration of the operation of an exemplary MEMs LiDAR based on [59].

Figure 2.28: A MEMs LiDAR works with microelectromechanically controlled mirrors for beam deflection. Using mirrors, different scan patterns can be driven. Typically, Lissajous patterns are used.

ToF principles: the pulsed ToF, Amplitude-Modulated Continuous Wave (AMCW) ToF, and Frequency-Modulated Continuous Wave (FMCW) ToF [59].

- The most widely used ToF principle is the pulsed ToF which sends a light impulse that is diffusely reflected on the target and is perceived by the sensors receiver diode. The time delay between the emission and reception of the very short impulse determined the range measurement.
- Compared with pulsed ToF LiDARs, AMCW ToF LiDARs use the intensity-modulated optical signal rather than the pulsed optical signal for sensing. To determine the range, the AMCW analyzes the phase difference between the transmitted modulated and received light. While the pulsed ToF principle is suitable for long-range distances, the AMCW ToF is better suitable for moderate distances and provides more accurate distance measurements.
- FMCW LiDARs emit a frequency-modulated light signal and compare frequencies of the reflected signal. While the time delay reveals the target's distance similar to the pulsed ToF principle, the frequency modulation makes detecting the Doppler effect possible. Consequently, in addition to the distance of the target, its relative velocity is determined. The FMCW ToF principle traditionally comes from the FMCW Radar signal processing. Applying the FMCW ToF principle to nm-wavelength signals requires expensive and dedicated hardware.

In the automotive sector and robotics, mainly pulsed LiDARs are utilized due to their simple and cost-effective ToF principle. Nonetheless, FMCW LiDARs were comparatively new when this manuscript was written and represent a promising alternative to the rather simple pulsed version since dynamic objects can be detected and distinguished in just a single measurement. [59]

Regarding the mechanical structure, LiDARs are distinguished between mechanical and solid-state LiDARs. Mechanical LiDAR sensors traditionally use a mechanical rotation mechanism to achieve a wide FoV scanning. As illustrated in Figure 2.27, mechanically rotating parts such as tilting mirrors deflect the light signal. While such LiDARs have a large FoV, the mechanical structure makes the sensor bulky and sensitive to vibrations. In contrast, solid-state LiDARs do

not have any mechanical components. At present, different topologies are used to implement solid-state LiDARs. The most widely used schemes are Flash-based, Microelectromechanical systems-based (MEMs), and Optical Phased Array-based (OPA) LiDARs:

- Flash LiDARs use a single light flash to illuminate the measured scene simultaneously. With a working principle similar to a camera flash, the sensor measures the distance of surrounding targets by recording the reflected light at different times. The main disadvantage of this technology is its sensitivity to retroreflective materials. [59, 61, 62]
- MEMs LiDARs substitute the mechanical part of the mechanical LiDARs with electromechanically controlled mirrors. By cascading multiple microelectromechanically tiltable mirrors to adjust the emission angle of the laser beam, 3D depth perception is possible, enabling miniaturized LiDAR sensors. Figure 2.28 illustrates the schematic structure of MEMs LiDARs. [59, 61]
- OPA LiDARs make use of a microarray of independent emitters. By controlling the timing between the signals transmitted by each antenna, the beam direction is controlled without a mechanical rotation [59, 62].

In the scope of this work, we use a traditional mechanical and a solid-state MEMs LiDAR sensor presented in Chapter 8.

Independent of the LiDAR principle, each laser beam provides the distance r to the reflecting target. Depending on the LiDAR topology, the polar (vertical) angle α and the azimuthal angle β as exit angles with respect to the LiDAR reference frame are determined for each laser beam. Computing the exit angles depends on the intrinsic calibration of the sensor. We compute the 3D coordinates of any measured point by converting the spherical coordinates r, α and β to Cartesian coordinates

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cdot \sin\alpha \cdot \cos\beta \\ r \cdot \sin\alpha \cdot \sin\beta \\ r \cdot \cos\alpha \end{pmatrix}. \tag{2.72}$$

However, the intrinsic calibration of the sensor is not perfect. As a consequence, deviations can occur to all three spherical coordinates. Those deviations are modeled as uncertainties of the measurement. Classical probabilistic approaches model the uncertainty by a Gaussian distribution. In contrast, in this work, we will account for those deviations by defining interval bounds on the uncertainty of the spherical coordinates as suggested in [51, 57]. The core idea is to inflate the spherical coordinates by the maximum possible errors Δ_r, Δ_α, and Δ_β, specified by the manufacturer. Thus, we obtain the intervals $[r] = [r - \Delta_r, r + \Delta_r]$, $[\alpha] = [\alpha - \Delta_\alpha, \alpha + \Delta_\alpha]$ and $[\beta] = [\beta - \Delta_\beta, \beta + \Delta_\beta]$. By extending (2.72) to interval operators and using the spherical interval parameters, we obtain a box $[\mathbf{p}']$ as illustrated in Figure 2.29. As stated in [51], the interval-based error model up to this point only considers imperfect intrinsic calibration and potential environmental impacts such as humidity and changing reflectance of surfaces. However, the initial footprint of the laser beam is not considered. In the scope of this work, we will neglect the minor impact of the initial footprint.

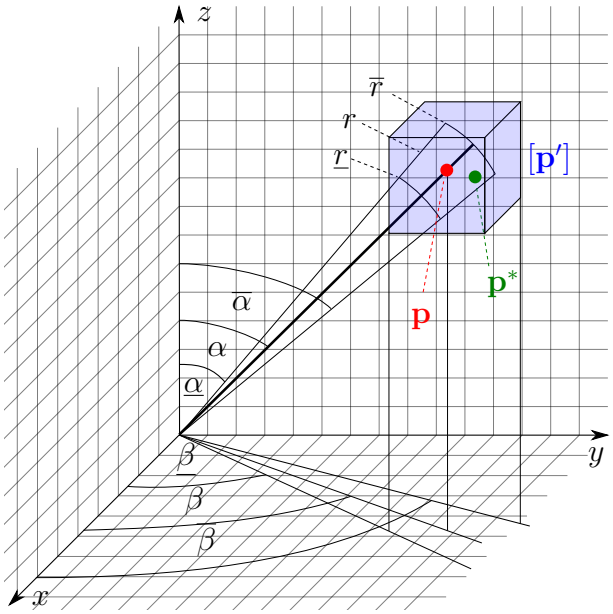


Figure 2.29: Transformation from spherical interval coordinates to the Cartesian coordinate frame leads to an inflated box. Hence, each measurement point $\mathbf{p}$ is inflated to box $[\mathbf{p}']$ that encloses the correct target position $\mathbf{p}^*$. The figure is adapted from [57].

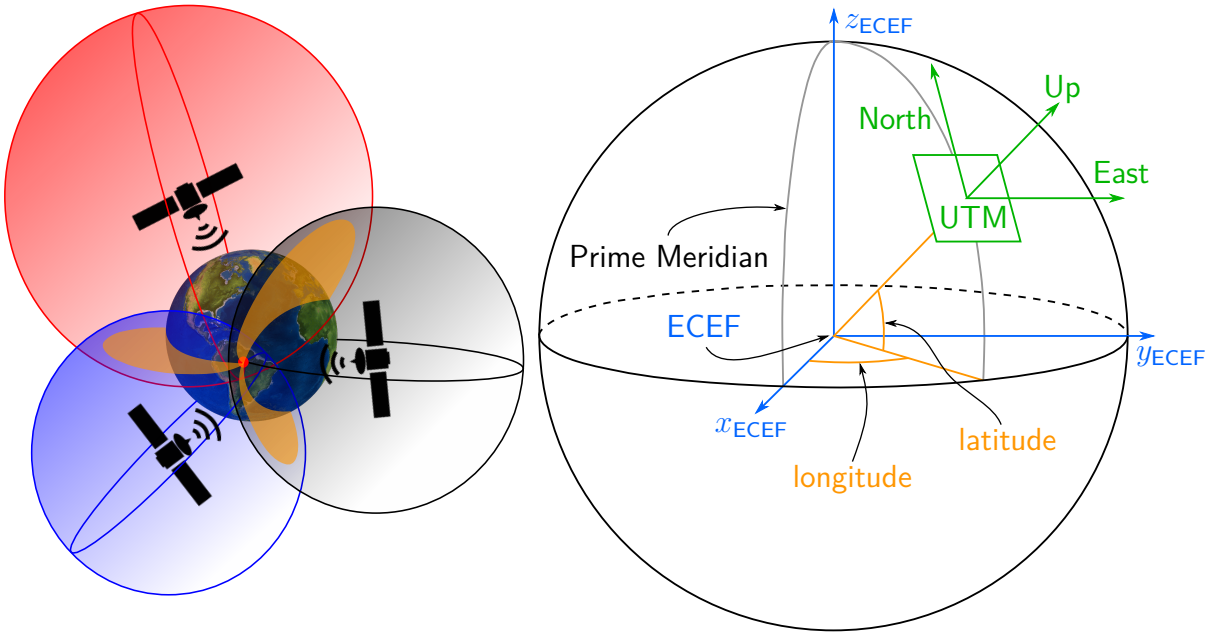
2.3.3 Global Navigation Satellite System (GNSS)

GNSS is a system for positioning and navigation on Earth and in space using signals emitted by satellites that orbit the Earth. The term GNSS generally covers all existing and future global satellite systems. To date, there are four different fully operational GNSSs:

- The Navigational Satellite Timing and Ranging – Global Positioning System (NAVSTAR GPS), often abbreviated by GPS, is the most widely used system developed and operated by the United States Space Force. Currently, 32 satellites exist, of which 24 are used in the core constellation to maintain good coverage.
- The Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS) is a Russian GNSS operated by Roscosmos. GLONASS also uses a constellation with 24 satellites, while 26 satellites are orbiting.
- The European Union Agency operates the European Galileo GNSS for the Space Programme (EUSPA) and the European Space Agency (ESA). Currently, there are 30 satellites, of which 24 are active and six are spare.
- The Chinese GNSS BeiDou is operated by the China National Space Administration (CNSA). BeiDou uses 30 satellites, while 35 are orbiting the Earth.

Furthermore, different supplementary systems exist that augment the GNSSs. [63]

Although different GNSSs exist, the basic principle of all systems relies on multilateralism based on the distances of the measured point to the observed satellites. Figure 2.30a illustrates the principle of satellite-based positioning. Each satellite broadcasts an electromagnetic signal that consists of multiple components. The physical layer defines the carrier frequency as the high-energy carrier signal on which further information is modulated (phase modulation). Two further data layers are modulated onto the carrier frequency that convey the information



(a) Trilateration of the receiver position.

(b) GNSS coordinate systems.

Figure 2.30: The general principle of how GNSSs works is depicted in Figure 2.30a. The GNSS-related coordinate systems are shown in Figure 2.30b. The GNSS receiver typically provides the spherical coordinates in the ECEF. We transform the position to a local tangential UTM coordinate system.

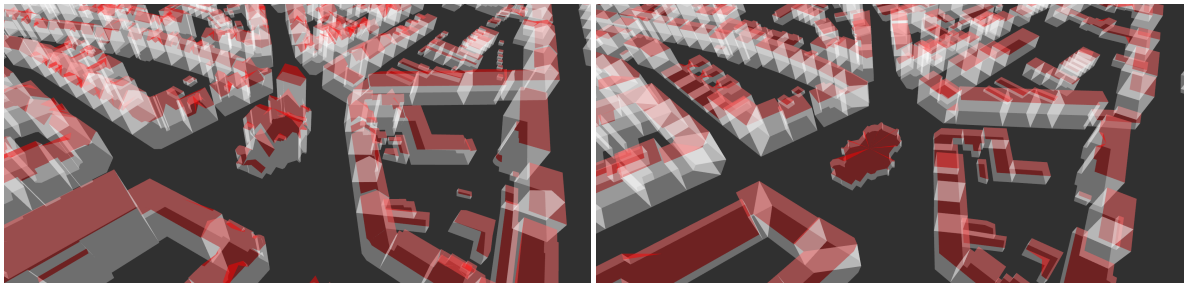
necessary for positioning the receiver. The ranging code layer provides information on the propagation time. It is a periodic modulated signal that is strictly synchronized to the satellite time system and the data messages to enable time synchronization of the receiver. The data link contains, among other things, the transmission time and satellite ephemerides (satellite position) [64]. Consequently, the receiver must first decode the information from the electromagnetic signal. The ephemerides decoded from the signals reveal the exact position of each satellite relative to the earth's center (geocentric coordinate frame). If the receiver employs a clock precisely synchronized to the GNSS time, the geometric distance to each satellite could be accurately measured by recording the run time required for the satellite signal to reach the receiver. Thus, here again, the distance measurement relies on a time of flight measurement similar to the LiDAR – with the difference that the signal travels a long distance and the time synchronization has to consider relativistic effects due to the speed and gravitational difference between the receiver and satellite. That means ranges to only three satellites would suffice since the intersection of three spheres, as illustrated in Figure 2.30a yields the three unknowns (e.g. latitude, longitude, and height). However, modern receivers apply a slightly different technique. As low-cost receivers use inexpensive crystal clocks, set approximately to system time, the time may be offset. As a result, the distance measured to the satellite also differs from the geometric range. Therefore, the measured quantities are called pseudoranges since they represent the geometric distance plus a distance correction resulting from the receiver clock error that defines a further unknown. Consequently, we now obtain four unknowns (latitude, longitude, height, and time offset) that are fully determined by four satellite signals instead of only three.

Up to this point, we only introduced the basic principle of GNSSs. In the scope of this work, we will also consider different global coordinate frames that we want to summarize in the following briefly. The receiver position is determined in a geocentric coordinate system, also known as the Earth-centered, Earth-fixed coordinate system (ECEF). The ECEF is a conventional 3D right-handed system. The fixation of the ECEF coordinate system (the origin with respect to the earth) is determined by the so-called geodetic datum. For example, the WGS84 is such a datum that is used for NAVSTAR GPS. The WGS84 models the earth as an ellipsoid, and the origin of the ECEF coordinate frame in WGS84 is the earth's center of mass. As shown in Figure 2.30b, the z-axis faces northward, while the x and y axes are in the equator plane. The x-axis faces to 0° longitude to the prime meridian, and the y-axis faces towards 90°E longitude. The GNSS receiver provides the position in WGS84 by longitude, latitude, and altitude coordinates (spherical coordinates) as shown in Figure 2.30b. Note that the altitude is referenced in the WGS84 earth ellipsoid. However, the building map we want to localize the vehicle is defined in the so-called Universal Transverse Mercator (UTM) coordinate system. In contrast to WGS84, which has its origin in the center of mass of the earth, the UTM coordinate system is a planar projected coordinate system defined on the surface of the earth as shown in Figure 2.30b. To obtain such a tangential coordinate system and minimize the projection's approximation error, the earth is divided into 60 UTM zones with different origins for the Cartesian UTM coordinate system. There are two possibilities to describe the orientation of the UTM coordinate system for a zone on the earth's surface: The East-North-Up (ENU) defines the x-axis to be oriented to the east. In contrast, the y-axis faces to the north, and the z-axis defines the height as illustrated in Figure 2.30b. An alternative orientation is the North-East-Down (NED) convention which we mention here only for completeness. To transform from the global position described in WGS84 to the UTM coordinate system in ENU convention, we utilize the Proj4 library [65] that performs the projection. [66, 67]

Since GNSS signals are sensitive to external effects, the position can be error-prone. As the signal passes a long distance through multiple atmospheric layers, the signal is refracted on the phase transition of different layers. Although the refraction can be compensated in the positioning, the refraction model can only approximate the complex dynamic behavior of atmospheric layers [68]. As this atmospheric effect cannot be fully eliminated with a single receiver, the error needs to be considered as further uncertainty in the positioning. Another problem of GNSS is the multipath effect: The transmitted GNSS signal may not always reach the receiver directly. If the receiver is placed close to large buildings or other objects, the signal may be reflected on the surface of the objects and may reach the receiver indirectly. Due to the reflection, the signal travels a longer distance, corrupting the position estimation. Especially in urban regions, the multipath effect may severely corrupt the position estimate [68]. This work aims to cope with the large uncertainty the GNSS positioning provides and improve the uncertain position with a local LiDAR sensor to building map association.

2.4 Building Maps

In this work, we introduce a novel building map-based localization pipeline. An efficient representation of the building is necessary to perform the localization in the building map. In



(a) LOD2 City Model. (b) OSM inflated to 3D.

Figure 2.31: The CityGML LOD2 model is shown in Figure 2.31a and the 3D inflated OSM building map is shown in Figure 2.31b

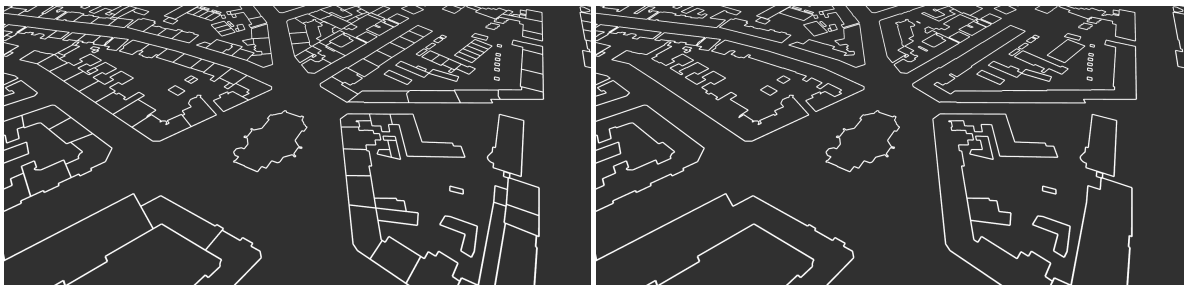
the scope of this work, we use the CityGML Level Of Detail 2 (LOD2) city model [69] and OpenStreetMap (OSM) [70]. While the LOD2 city model is only available for dedicated cities, OSM provides worldwide geospatial data.

A portion of the LOD2 city model of Hanover is shown in Figure 2.31a. The generalized 3D model of the buildings is based on cadastral information and data from airborne laser scanning. Only dedicated cities provide such publicly available building maps. For the two author-collected datasets recorded in Hanover, the LOD2 maps are available and officially maintained and distributed by the city government's planning and urban development department. The main advantage of the LOD2 city model is its precision since airborne laser scanning provides highly accurate data. Nonetheless, the LOD2 model generalizes complex facade shapes to simple planes, introducing generalization errors in the building map. Since also cadastral data is considered within the LOD2 models, the uncertainty is comparatively small, with standard deviations between 2 to $10\,\mathrm{cm}$. The LOD2 map also provides generalized roof shapes as shown in Figure 2.31a. However, in the scope of this work, we assume the vehicle to move on the streets so that the roofs are generally not captured by the sensors and therefore do not contribute to the localization.

In contrast to the LOD2 model, OSM is a publicly maintained map where everyone can contribute to the map. Consequently, the OSM database provides maps worldwide and combines different information like street names, building numbers, and restaurant locations. Although the maps provide rich environmental information, OSM only provides 2D building footprints. However, for some buildings, the height information is also available. Figure 2.31b shows a 3D inflated OSM building map of the same region as in Figure 2.31a. Note that the roofs are visualized flat since no information on the roof shape is available in OSM.

One of the main problems with OSM data is its reliability. Although the OSM database is frequently updated and corrected, different contributors are equipped with different measuring instruments with different accuracies. As a result, generally, it is not possible to quantify the uncertainty of the map. However, we found that we could reliably bound the maximal error motivating the use of interval-based approaches for the maps we used for the different datasets.

Since neither the LOD2 nor OSM provides information on the ground terrain, the building map-based localization cannot constrain the elevation of the vehicle. Furthermore, to ensure the global applicability of our approach by maintaining compatibility with OSM that only



(a) All walls available in the building maps. (b) Only visible walls are considered in queries.

Figure 2.32: The building footprints also contain hidden walls between directly connected buildings, as shown in Figure 2.32a. We only consider outer walls for map queries, which are visible as shown in Figure 2.32b.

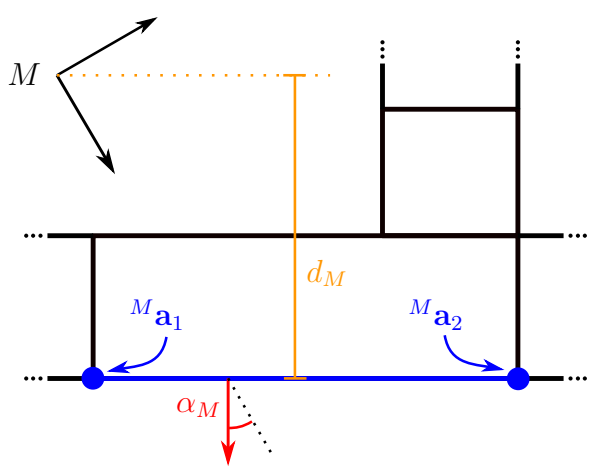
provides 2D maps, we perform the localization in 2D. We only use the building footprints illustrated in Figure 2.32a.

The footprint of buildings also contains hidden walls that the sensors cannot see. Such hidden walls are, for instance, between connected buildings within a series of buildings. Especially in urban environments, such dense building structures are ubiquitous, as shown in Figure 2.32a. As we are only interested in the visible walls, we only consider the outer building walls in a map query. The footprint submap that only contains the visible walls is shown in Figure 2.32b.

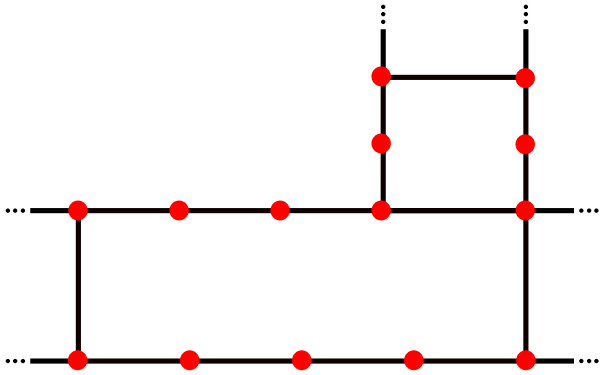
Consequently, we assume a map consisting of building footprints with uncertainties for which we can give an upper and lower bound. A building is modeled by multiple facades where each facade forms a line segment in the map and connects the corner points of a building. That means each building is represented by a closed polygon in the map, defined by a series of corner points. A line segment is defined by a pair of building corner points ${}^{M}\mathbf{a}_1$ and ${}^{M}\mathbf{a}_2$ described in the map frame M as visualized in Figure 2.34. We further represent the line by its implicit form

$$\mathbf{n}_M^T \cdot {}^{M}\mathbf{a} - d_M = \begin{pmatrix} \cos(\alpha_M) & \sin(\alpha_M) \end{pmatrix} \cdot \begin{pmatrix} {}^{M}a_x \\ {}^{M}a_y \end{pmatrix} - d_M = 0. \tag{2.73}$$

The orientation of the facade is described by the angle α_M, while the distance of the line to the origin of M is defined by d_M as shown in Figure 2.33a. As we assume interval uncertainty for the facades, each corner point inflates to a box, and the line parameters also inflate to intervals. Consequently, the i-th facade in the map is fully described by $F_i = \{[{}^{M}\mathbf{a}_1], [{}^{M}\mathbf{a}_2], [\alpha_M], [d_M]\}$. To ensure efficient map queries like radius or nearest neighbor searches, we employ an additional KD-Tree representation of the building footprint map. Since the KD-Tree organizes a set of points for efficient search, we need to represent the building map by a set of points. Therefore, we homogeneously subsample points from the 2D projected facades, taking the start and end-points into account as illustrated in Figure 2.33b. Since the map queries are performed from local measurements provided by the sensors, we only represent the outer visible buildings walls, omitting hidden walls, in the KD-Tree representation. Further, for each point, we also store the information to which facade it belongs. During the radius search, points near the search point are selected. As each point is associated with the corresponding facade, we determine the facades associated with the points in the vicinity.



(a) Visualization of the facade parameters.



(b) Exemplary sampling points to build the KD-Tree for efficient map queries. For each sampling point, we store the association to the original facade in the map.

Figure 2.33: Exemplary map to illustrate the facade parameters (Figure 2.33a) and the sampling of points for the KD-Tree (Figure 2.33b).

Figure 2.34: The convex polygon hull defines the border of the map. Black-filled polygons visualize the buildings. The map corresponds to the evaluation dataset KITTI 0018.

Although the building map can be arbitrarily large, our localization pipeline relies on closed building maps. As shown in Figure 2.34, we choose the convex polygon hull as the map's border. We assume the vehicle is only operated in the closed region inside the map.

The OSM and LOD2 city model is defined in UTM. Since we only use datasets acquired in Germany, we stick to the 32N zone. Nonetheless, the corner points of the buildings have large coordinate values since the origin of the 32N zone is several $100\,\text{km}$ away. That is why we define a local coordinate system with the same orientation as the UTM frame but only offset in translation to a nearby point. This point defines the origin of our map frame M. The localization is performed in M.

3 State of the Art

Robot localization has been researched for decades, and many approaches have been proposed. The goal of this chapter is to provide a summary of the historical development of robot localization algorithms. Since in the HyPaSCoRe Localization a hybrid interval-probabilistic visual odometry module utilizing a SLAM-graph is introduced, we also provide for the interested reader State of the Art visual odometry and SLAM approaches in the appendix Chapter B.

Three types of robot localization problems are distinguished in the literature – namely *robot tracking*, *global localization*, and *kidnapped robot problem*. Robot tracking assumes the robot's initial pose to be known. However, this information can be corrupted by noise and uncertainties. In contrast to the global localization problem, the initial pose uncertainty is rather small. Hence, the tracking problem is local since the uncertainty is small and the initial estimate is close to the correct pose.

Unfortunately, this initial information is not always well known. Especially in urban canyons, global positioning systems may provide very inaccurate pose estimates due to occlusions and/or multi-path effects. In the worst case, no global information on the robot's initial pose is available. While tracking approaches rely on the assumption that the pose error is small, for the global localization problem, this assumption is not always valid, as the initial pose uncertainty is rather large.

The kidnapped robot problem is a variant of global localization, where the robot can get kidnapped and moved to another location during the operation. In the scope of this work, we only focus on robot tracking and global localization, as those types of problems are the most relevant ones in the context of autonomous driving. In the literature, a large variety of different approaches have been suggested to those types of problems. The approaches differ regarding the map, the sensors used, and the uncertainty representation of the involved variables. We introduce and discuss relevant State of the Art probabilistic, interval-based, and hybrid robot localization approaches in the following.

3.1 Probabilistic Approaches

Probabilistic approaches model the observations and states by probabilities. Probabilistic localization algorithms are variants of Bayes filters. The application of Bayes filters to the localization problem is called Markov localization. Markov localization approaches mainly differ concerning the underlying state space representation [71, 72]. While the Extended Kalman

Filter (EKF) represents the state space by the first and second moment of the belief, grid-based techniques apply histogram filters. The most prominent Monte Carlo Localization applies the particle filter approach to solve the localization problem. Maximum Likelihood Estimation (MLE) approaches have recently gained more popularity due to mature numerical optimization techniques. The Gaussian assumption enables the formulation of the MLE approach as a least-squares problem that robust optimization techniques can solve [23, 29].

3.1.1 Extended Kalman Filter

The EKF is one of the most classical probabilistic state estimation approaches. Leonard and Durrant-Whyte present in [73] one of the first applications of the EKF to the localization problem. The authors first match the locally observed geometric beacons to the beacons on the map. Utilizing the matches, the authors determine the robot's pose on the map with an EKF that models the pose as the state parameters. The prediction step uses the control input of the robot. The locally observed beacons are extracted from the sonar data to perform the update step. As the authors assume that to accurately know the robot's initial pose, the matching of the local sensor data to the map beacons boils down to the nearest-neighbor association. By applying the EKF for each iteration step, the authors successfully track the robot in the simple artificial map.
Iocchi et al. [74] extend the EKF-based tracking to line-like and circular structures in the environment. Therefore, the authors introduce a Hough Transformation [75], aggregating the locally measured point into the Hough space. The sets of points that form line-like structures generate local maxima in the Hough space. The map features are also transformed into the Hough space which means lines in the map become points in the parameter space. As the authors assume to know the initial pose, the local maxima are closely located to the feature points in the Hough space. Hence, the authors can match the local data to the map features by employing a simple nearest-neighbor association. Using the matches, the authors correct the predicted robot pose in the EKF.
Teslic et al. [76] use the EKF to localize a four-wheeled mobile robot equipped with encoders for the wheel and a 2D LiDAR. The authors assume the robot moves in a structured environment with well-defined and mapped line-like walls. In the prediction step, the pose estimate is determined by simulating the robot's kinematic model. The pose is then corrected by minimizing the difference between the local and global line segments. Local line segments are matched to the global map lines in a nearest-neighbor fashion. Hoang et al. [77] present the EKF-based localization with additional sensors such as a compass and an omni-directional camera. Like the classical EKF approach, the authors use wheel encoders to predict the robot's pose. In the correction step, the authors improve the state estimate by fusing the compass data and omni-directional camera images into the computations. While the compass improves the orientation estimate of the robot, the authors extract an artificially placed red-colored landmark in the arena that is tracked in the image. Based on where the red landmark is detected in the omni-directional image, the orientation of the robot is corrected.
All localization approaches that were presented are only applicable to the robot tracking problem. Generally, linearized Gaussian techniques mainly work well if the uncertainty of the initial estimate is small enough. That is why the tracking approaches are well modeled

Figure 3.1: Pose candidates in a building map-based localization. The blue dots mark the valid and the red invalid candidates. [78]

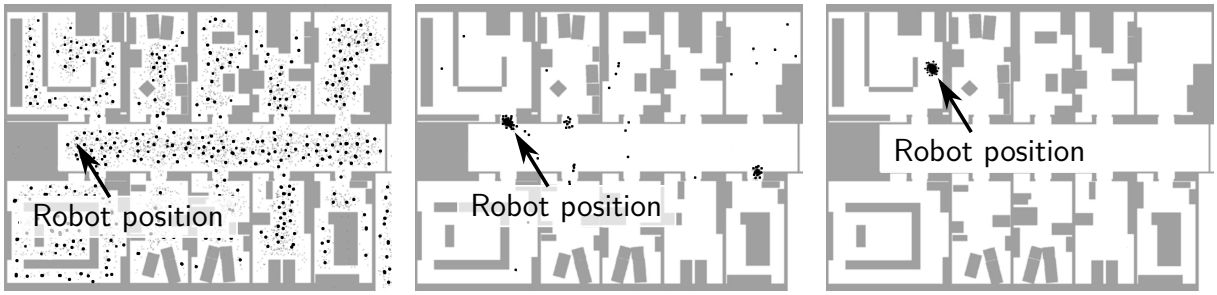
by, for instance, unimodal probability distributions. Regarding global localization, unimodal distributions are usually inappropriate. As a result, classical EKF approaches are not generally suitable for the global localization problem. However, there exist extensions of the EKF that represent a belief by multiple Gaussians, like the Multi-Hypothesis Tracking (MHT), that can be employed to solve the global localization problem.

Landsiedel et al. [78] present an MHT global localization approach that performs chamfer matching between the locally seen building outlines and corresponding map sections. As input data, the authors use a LiDAR scanner to extract planes projected to two-dimensional lines. Matches between the building edges from the sensor data and the 2D building map are computed by a template matching procedure. Appropriate candidate poses are selected and refined by a chamfer matching method similar to [79]. Figure 3.1 depicts the exemplary result of pose candidates. The approach can become very heavy in computation, depending on the number of tracked candidates.

3.1.2 Grid-based Localization

The grid-based localization is a metric variant of Markov localization. Moravec and Elfes initiated the idea to use a certainty grid map for obstacle representation [80]. However, Burgard et al. present in [81] one of the first approaches that use the same grid map representation for localization. Therefore, the authors invert Elfes' and Moravec's idea by constructing a position probability grid where each grid represents the posterior probability that the grid includes the robot's current position. Similar to the EKF-based localization, the grid-based approach also consists of two steps: First, the probability grid is shifted according to the robot's odometry measurements, taking dead-reckoning errors into account. Second, for each cell in the grid, the position probability is determined by combining the likelihood of the local reading – supposed the cell is the robot's current position in the map – with the likelihood already stored in the cell.

Fox et al. extend [81] in [82] by improving the robustness of the approach for highly dynamic and densely crowded environments. Therefore the authors augment the grid-based localization with



(a) Initial particle set. (b) After 2 m robot motion. (c) After moving into the room.

Figure 3.2: Global localization of a mobile robot using MCL with 10000 particles. The images are adapted from [85]. The further the vehicle moves, the denser the particles converge to the correct vehicle position.

a filtering technique that updates the position probability density using only those measurements with high likelihood produced by known objects in the map. The presented approach was successfully applied to tour-guide robots in the *Deutsches Museum Bonn* and the *National Museum of American History*.
The grid-based localization is capable of solving the tracking and global localization problem. The major drawback is the computational burden and memory consumption. While a high resolution of the grid leads to a higher accuracy of the localization estimate, the memory consumption and the computation time rise. A trade-off between accuracy and runtime is necessary.

3.1.3 Monte Carlo Localization

The MCL was first presented by Dellaert et al. in [83] and represents an important milestone in probabilistic robotics. As a classical filtering approach, the particle filter has a prediction phase, in which the motion model is applied to each particle, and an update phase, in which the particles are weighted based on the likelihood of the pose given the local measurements and resampled from the weighted set. While Dellaert et al. present the basic version of the MCL, in the following years, many different variants and extensions have been published. We only mention a few selected variations of the MCL approach. For instance, Fox presents in [84] an extension of the basic MCL algorithm that adapts the number of the drawn particles based on the approximation error. The approximation error is modeled by the Kullberg-Leibler distance (KLD). If the KLD measure is small, the particles are concentrated on a small part of the state space, and the number of particles can be reduced in the MCL approach. Otherwise, a large sample set is necessary if the state uncertainty is high. Thrun et al. [85] further robustify the MCL to a Mixture-MCL by integrating two complementary ways of generating samples in the estimation. Figure 3.2 illustrates the basic principle of the MCL.
Since the MCL represents a powerful tool, also recent works integrate the particle filter into their localization procedure. Hentschel and Wagner [86] use MCL to localize their autonomous vehicle in an outdoor environment using OpenStreetMap (OSM) data. The authors use MCL only for tracking since they receive global localization information from GPS data. While the GPS position fix is Kalman filtered with the vehicle's wheel odometry and IMU data, the

resulting filtered pose is integrated into the MCL by adding a small number of samples drawn from a Gaussian distribution centered at the Kalman Filter pose. The authors demonstrate they can navigate an autonomous vehicle using the localization estimate.
Using visual odometry, Floros et al. [87] localize a robot on the road network. Using a local history of previous poses, the trajectory is matched to the road graph by applying chamfer matching. The localization scheme is structured in an MCL framework where each particle is weighted based on the matching result of the local trajectory to the road graph. The approach needs a rough initial GPS position.
Ruchti et al. [88] classify laser scans into the road and non-road measurements. The classification is used in a corresponding observation model to weight the particles of an MCL based on OSM data.
Yan et al. [89] globally localize a robot on OSM data using a 4-bit semantic descriptor. The descriptor encodes information about the visibility of road intersections and building gaps. Based on OSM data, the authors first condense the OSM data to a set of 4-bit descriptors by computing the descriptor for all potential positions and orientations in the map. Further, the authors perform a semantic segmentation for each 3D LiDAR scan and compute a local descriptor. To localize the vehicle, the authors combine the expected and real observation represented by descriptors to define the observation model for the MCL. Each particle is weighted based on the hamming distance of the map descriptor and the local descriptor.
Chen et al. [90] perform global localization in a self-built hybrid map that consists of visual keyframes and an occupancy map using camera data and LiDAR. A global image descriptor matching is applied to search the referenced keyframes according to the current visual observation and is used as the observation model of an MCL approach. After global localization with the image descriptor matching, LiDAR-based tracking is maintained.
Chen et al. [91] perform MCL using range images generated from real LiDAR scans and synthetic renderings of a mesh map. The difference between the range images is used to formulate the observation model for the MCL. The authors show that a high amount of particles is necessary to localize the vehicle successfully. That is why the authors report high computation time before convergence.
Zhang et al. [92] augment the MCL approach by a particle swarm optimization. In contrast to the classical MCL approach, the authors suggest optimizing the particles with a least-squares approach. Therefore, a fitness value for each particle needs to be determined, increasing the computational burden. However, due to the optimization step, the authors report that fewer particles are necessary to let the estimation converge to the correct solution. While in conventional MCL methods, each particle individually tracks the pose as the robot progresses, in the proposed approach, particles in each sub-swarm track the robot's pose in the corresponding hypothesis. Clusters of the most likely particles generate sub-swarms. When a new observation is available, particles in a sub-swarm are optimized to move towards the maximal likely region. By performing normalization and weighted resampling as in the classical MCL, unlikely pose hypotheses and sub-swarms are eliminated gradually from the global localization results, and the remaining particles converge to the correct pose. This approach combines MCL with techniques of MLE approaches that are introduced in the next Subsection 3.1.4.
Recent works, as presented in [93] and [94], use learned observation models to perform particle weighting within the MCL framework. Chen et al. propose a neural network-based observations

model that computes the expected overlap of two 3D LiDAR scans. The learned model predicts the overlap and the yaw angle offset between the current sensor reading and the virtual frame generated from a pre-built map. The predicted overlap score is integrated into the particle weighting in the MCL framework. Zhou et al. combine the MCL with matching ground-level images to 2D cartographic maps such as OSM provides. The matching is based on a learned embedded space representation linking images and map tiles. The matching score is piped into the MCL as a weighting mechanism of particles.
The core of MCL-based approaches is the observation model that determines the weighting mechanism of the particles. Note that the mentioned MCL approaches differ in the observation model while the rest of the particle filter algorithm stays identical. While for instance, Hentschel et al. [86] use the classical beam-end model for the observation model, Yan et al. [89] propose an observation model based on a 4-bit semantic descriptor. Machine learning-based observation models are presented in [93, 94]. The MCL is one of the most popular localization algorithms in the robotics community due to two facts: Implementing the MCL algorithm is comparatively easy, and, as it can approximate nearly any distribution, it is also one of the most potent ones [18]. Hence, the particle filter approach applies well to the tracking and global localization problem. However, one of the significant problems of MCL approaches is that the quality of the localization solution heavily depends on the number of particles. If the uncertainty is very large, many samples may be required to cover the solution space. As a result, updating and correcting a large number of particles leads to more computation. That also means that a trade-off between accuracy, convergence speed, and runtime has to be made for the MCL, similar to grid localization. Another problem the MCL has to cope with is the wrong convergence of the method caused by an unfortunate sequence of random samples [95]. As a consequence, MCL approaches lack integrity, and often results are not repeatable, as the sampling is random.

3.1.4 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) approaches seek to find the most likely solution within the solution space. Therefore, the probability distribution within the solution space needs to be determined. This probability distribution is generally defined by the probability distributions of the local measurements, those of the given map, and the function that maps the measurements to the solution space. The probability distribution of the set of poses is determined by propagating the probabilities from the measurement space (the local measurements and the map) to the solution space (the pose of the robot). Unfortunately, this propagation step is difficult for arbitrary probability distributions and non-linear mapping functions. To cope with that problem, different approaches and assumptions have been introduced in the literature, briefly summarized in the following.
In [96], Olson presents a maximum likelihood map matching method, where the robot location is determined by the pose that maximizes the agreement between the local map retrieved by range data and the given map. Therefore, the author introduces a search strategy to locate the most likely pose. First, an arbitrary nominal position of the robot is considered an initial position that provides a likelihood to compare. Then, the pose space is divided into rectilinear cells. For each cell, the likelihood of the pose is determined, and only good cells are further analyzed. The likelihood of a pose is determined by a map similarity measure based on the locally seen

features and features in the map. The branch-and-bound strategy determines the most likely pose for a given search space. The presented method was successfully applied to the Sojourner Mars rover. The main advantage of this method is that an arbitrary probability distribution in the solutions space is well represented by the cells. This approach is very similar to grid localization as it maintains subparts of probability histograms. However, on the downside, the branch-and-bound search algorithm leads to a high computational burden depending on the defined search space.

MLE approaches dealing with arbitrary probability distributions must overcome many hurdles, often leading to a high computational burden. To avoid this problem, a very typical probabilistic prerequisite in the literature has proven to be an elegant solution: When all major measurement disturbances are eliminated – that means if there are no systematic errors in the measurements – the remaining errors are formed from many small error sources. According to the *Central Limit Theorem*, the probability distribution of the sum of N small independent random variables tends to the normal distribution when N increases. Hence, the Gaussian assumption can be a good approximation for many independent measurements without systematic errors. Assuming a normal distribution for all the measurement errors, it is mathematically shown in [10] that the MLE becomes a least-squares method. This is indeed good news for probabilistic approaches, as there exist effective and robust least-squares optimization methods [23, 29, 97] that can be employed to solve the MLE formulation of the localization problem.

One of the most popular scan-matching – or more generally speaking scan-registration – methods is the Iterative Closest Points (ICP) algorithm [98] that exploits the Gaussian assumption and formulates the matching problem as a least-squares problem. The registration method can also localize the robot in a given environment, as presented in [99]. Different variants of the ICP algorithm, such as [100–102], mainly differ in the cost function that is minimized. Nevertheless, the optimization problem is always formulated as a least-squares problem so that standard optimizers can be used.

Vysotska and Stachniss [103] localize a robot with a LiDAR in building maps retrieved from OSM data. The authors extend a standard pose graph-based SLAM formulation by relating dedicated nodes of the pose graph with existing building map information by including prior information to the pose graph determined by the localization procedure. The authors first filter the range scans to localize the robot in the building map so that most non-building objects are removed. Then, the filtered scan is matched to the building map as illustrated in Figure 3.3. Therefore, the authors apply the standard ICP algorithm. As a result, the proposed approach cannot be applied to global localization since the ICP approach needs a good initial guess for the pose. The authors initialize their method with a manually specified first pose or based on GNSS data. By integrating the localization as prior information to the pose graph, the authors show that they can stabilize the SLAM-graph and even detect inconsistencies in the building maps.

Similar to [103], Boniardi et al. [104] propose a scan-to-map-matching method based on Generalized ICP [100] to localize an indoor robot in architectural CAD drawings. In contrast to [103] and [104], that integrate the localization information as prior data to the pose graph, Wilbers et al. [105] directly integrate third-party maps as prior knowledge about the states of the detected landmarks. The authors consider detections of pole-like objects as landmarks and match them to an existing third-party map. To perform the matching, a variant of the ICP

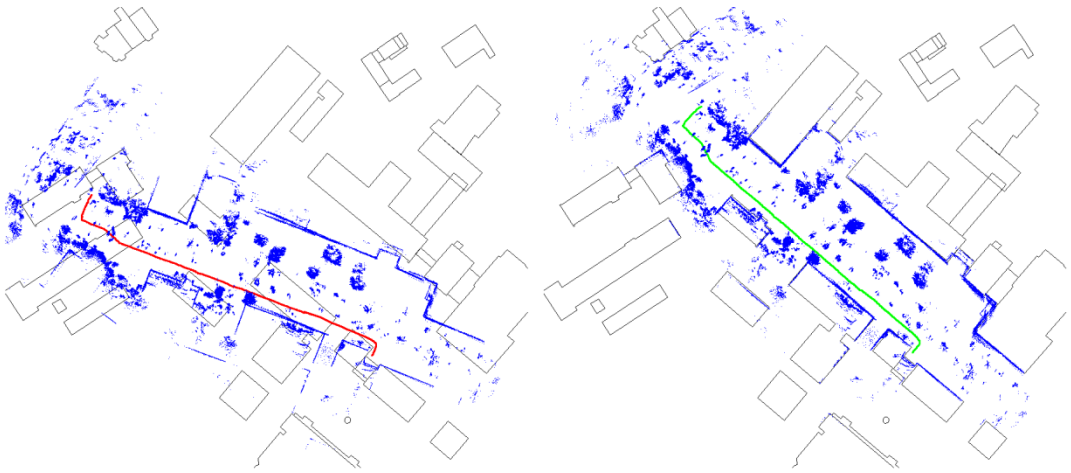


Figure 3.3: OSM data is used to align the robot's trajectory. The left image shows the pointcloud before and the left after alignment. [103]

approach is used to search for associations between landmark detections and mapped landmarks. All ICP-based map association approaches are inherently local since ICP only performs a local search. As a result, [103–105] can only solve the robot tracking problem as they have to rely on a good initial pose estimate.
Ratz et al. [106] and Cho et al. [107] suggest probabilistic approaches using unique descriptors for matching. While distinctive descriptors represent the map, local descriptors are computed and matched to the map database similar to [89]. Ratz et al. present a global localization algorithm that uses only a single 3D LiDAR scan at a time. First, the authors extract segments for which descriptors are computed based on a neural network. The segment descriptors are then matched against a database representing the map. After a geometric consistency test, a least-squares problem is solved to obtain the 6 DOF pose. The authors do not comment on multiple feasible solutions in the case of strong symmetries. Cho et al. generate a descriptor database for an OSM map based on the distances to buildings from arbitrary locations at a regular angle. Further, the authors determine a descriptor for the local LiDAR scan by taking the shortest distances to building points from the current location at a regular angle into account. The authors can localize the vehicle by comparing the local descriptor to the database. However, the presented method only keeps track of the most likely solution. The approaches work well if the parts of the map are well distinguishable. In the case of strong symmetries, the methods do not perform well, as multiple localization hypotheses are not tracked.

3.2 Interval-based Approaches

Interval-based approaches model the sensor and pose estimation uncertainties by intervals. While probabilistic methods propagate probability distributions from the measurement space to the state space, interval-based techniques perform the propagation utilizing interval arithmetic. The philosophy of how the same localization problem is solved differs significantly. Probabilistic approaches do not aim to enclose the correct solution. Instead, the goal is to determine the probability distribution in the state space that provides the information of how likely a state

represents the true state given the measurements. In contrast, interval-based approaches seek to provide an enclosure represented by interval sets that guarantee to enclose the true solution if and only if all assumptions are fulfilled. While probabilistic approaches are attracted by the most likely solutions, interval methods seek to keep track of all feasible solutions by dismissing infeasible parts.

Kieffer et al. [108] present one of the first interval-based static localization methods with ultrasonic sensors for structured 2D indoor environments. The prior map consists of oriented segments (walls) that describe the landmarks. The authors assume that within the mapped environment, there are no other unmapped obstacles that the sonars can perceive and that the vehicle must always be located within the map. Based on those assumptions, the authors formulate localization tests that qualify a set of poses as feasible, infeasible, or undetermined. The set of feasible poses for a given map and a given set of sonar readings is determined by applying SIVIA: Starting with an initially large pose box, the set of poses is recursively divided into subpavings individually tested by the localization tests. If the tests qualify a subpaving as infeasible, this part of the solution will be omitted. It becomes part of the feasible solution set if it is qualified as feasible. Otherwise, if the subpaving is apt to contain the feasible set of poses partially, the subpaving is further bisected until the minimal interval width is reached. This approach makes global localization possible, as multiple disconnected feasible solutions can coexist in the estimation. However, the computation time for this testing-based SIVIA approach is high, depending on the map size. The same authors extended their approach in [109] to a tracking method, where they also incorporated the previous estimate and the odometry data into the interval-based localization procedure. While initially, the same localization has to be applied to localize on the map globally, the tracking problem is solved by an update and correction procedure similar to the Kalman Filter architecture. In the update step, the previous set of feasible poses is updated by the odometry data taking interval uncertainty into account. Hence, the size of the interval estimates gets shifted and inflated. In the correction step, the same SIVIA approach is applied. In [110, 111], concrete applications that use the presented approach are shown. In [112], the authors extend the localization approach to a robust method that can deal with a defined number of outliers by a q-relaxation. However, the method can only cope with a small number of outliers. That means unmapped obstacles in the environment will lead the technique to fail. Further, the proposed method only deals with very few measurements and therefore is not well suited for laser scanners.

Gning and Bonnifait [113] present a localization approach that fuses dead-reckoning data derived from wheel encoders and the angle of the driving wheel with differential GPS data. The authors determine the vehicle's displacement using the wheel encoders, driving wheel orientation, and the motion model. Therefore, interval errors are estimated for the encoders and are propagated to the displacement estimate by formulating the problem as a Constraint Satisfaction Problem (CSP). The CSP is solved by utilizing contractors. This approach makes the error propagation very fast but possibly more pessimistic than the SIVIA approach presented in [108]. The GPS data is used to initialize the pose estimate. In [114, 115], the authors extend the set of sensors with an additional gyro and compare their approach with an EKF approach. The authors show that the interval-based approach is more pessimistic but provides consistent estimates. The EKF does not guarantee that the estimated error ellipse encloses the correct solution and occasionally loses track of the correct vehicle pose. Lambert et al. [116] further

evaluate the odometers, gyro, and GPS-based localization with outdoor experiments with a car equipped with embedded processors. The interval-based localization approach is processed in real-time and is compared with a particle filter approach. The authors come to a very similar conclusion, as stated in [114, 115] that the particle filter is more accurate but tends to diverge in the case of biased GPS measurements. This cannot happen to the interval approach as long as the error bounds are satisfied. Nonetheless, the authors only consider proprioceptive and absolute measurements. Clérentin et al. [117] extend the proposed constraint propagation approach to exteroceptive measurements. The robot has an omnidirectional vision system, a 2D LiDAR, and two odometers. The authors seek to localize the robot in a map with high-level primitives like corners and edges in an indoor environment. Therefore, the authors first extract locally seen primitives from the range data and the omnidirectional images. Then, in contrast to [108], the authors explicitly associate the locally seen primitives to the mapped primitives by taking the constellation into account. Each matching possibility leads to a new track represented by a subpaving. If a matching turns out to be incorrect, the constraint propagation approach will lead to an empty set for the pose estimate, and the track will be omitted. Hence, only feasible matchings remain and are tracked.

Sliwka et al. [53, 118] propose a robust version of an interval-based localization approach applying the image contractor. While the previously presented approaches assumed a structured environment that can be modeled by line segments, the authors seek to localize an underwater robot in an unstructured marina. As a result, the approximation of the environment with line segments becomes inapplicable, so the authors propose a binary image representation of the environment. While the marina walls are represented by ones in the regular image grid, non-wall parts of the map are marked with zeros. An imaging sonar sensor provides the local data used for localization on the map. For a given set of poses, the authors propagate each sonar beam measurement to the map frame and contract each local measurement to the marina wall applying the image contractor. In a back-propagation step, the contracted measurements lead to the contraction of the robot pose estimate. However, the sonar readings may be corrupted by outliers. That is why the authors use the q-relaxed intersection approach that excludes inconsistent measurements from the contraction. The authors successfully apply the approach to real data acquired in a Costa Brava marina. The main problem of the approach is that for the q-relaxed intersection, the number of outliers needs to be determined. Unfortunately, this information is not a priori available.

Langerwisch et al. [119] present an approach for localization in a structured indoor environment using wheel odometry and a 2D LiDAR. The authors represent the map by line segments defined by their start and end points similar to [109–111]. However, also the uncertainty of the line segments is considered in the map representation. While the odometry data is used for the prediction step, as previous works also did [112, 113, 117], the update step takes the local 2D LiDAR measurements into account. Therefore, the authors subdivide the initial pose estimate into subsets, representing the initial set by a subpaving. The local measurements are transformed from the sensor frame to the map frame for each subset. Due to the uncertainty of the pose subsets, the uncertainty of the local measurements gets further inflated in the map frame. For each transformed measurement, the authors determine the hull among the intersections of the measurement with map line segments. Hence, the transformed measurements are contracted to that hull. The authors can propagate this information to

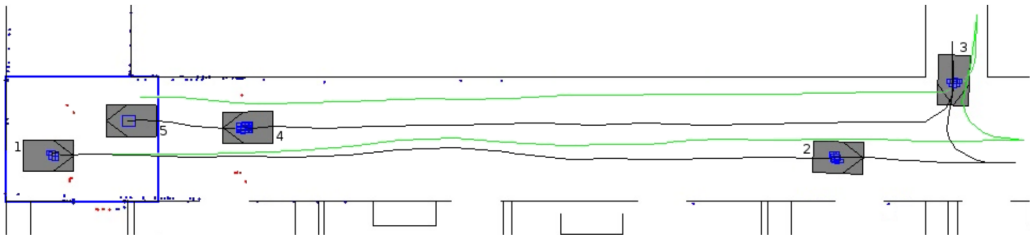


Figure 3.4: Experimental results of indoor interval-based localization. The estimated mean trajectory is colored black, the green trajectory depicts pure wheel odometry, and the interval boxes are painted blue for five situations. [119]

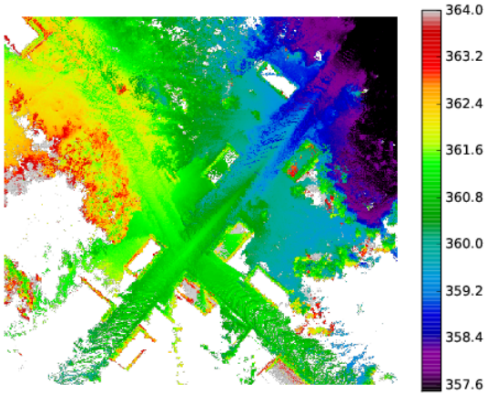
the pose by applying a forward-backward contractor. However, unmapped objects in the environment and erroneous measurements may lead to outliers. The authors account for that problem by introducing a q-relaxed intersection in the update step when applying the forward-backward contractor similar to [53, 112]. Langerwisch et al. successfully apply the tracking algorithm to an indoor robot that is operated in real-time as shown in Figure 3.4. On the downside, the initial pose estimate must be reasonably small to make the computational burden feasible. Further, the authors only account for a fixed number of outliers in the local sensor data. If the number of outliers exceeds this limit, the pose estimate may become invalid and is not guaranteed anymore.

Guyonneau et al. [120] combine the approaches presented in [53] and [119] to solve the global localization problem in indoor environments. While the authors use the same grid-based image-like representation of the map as presented in [53], the interval-based localization procedure is very similar to [119]. Additionally, the authors compare the interval-based method with a classical MCL approach. The authors report that the interval approach initially needs more computation time than MCL. However, the computation times are comparable when the domains are significantly reduced. On the one hand, the authors point out that MCL does not provide deterministic results, so the MCL tends to converge to an incorrect solution for certain runs. This does not happen with the interval approach, which – on the other hand – is more pessimistic.

Desrochers et al. [121] perform global localization using a terrain model as a map and a LiDAR. By robustly contracting the robot's pose to the feasible and consistent portions in the terrain model (cf. Figure 3.5) by taking the local LiDAR scans into account, the authors can localize the robot without the necessity to move the platform. On the downside, the approach is comparatively slow, and a good terrain model is necessary to obtain good results.

Kenmogne et al. [122] apply an interval-based localization approach to an unmanned aerial vehicle (UAV). The UAV is equipped with a camera, IMU, and a barometer. In the environment, mapped landmarks are placed that can be tracked by the camera. The authors localize the vehicle by matching the captured landmarks in the images to the mapped landmarks. Therefore, the authors exploit the Perspective-n-Point (PnP) constraints and build a forward-backward contractor that contracts the pose taking the landmarks and the local observations in the image into account. The contractor is applied in a SIVIA algorithm. The authors show that the interval-based approach occasionally leads to empty sets caused by inconsistent measurements. If such an event occurs, the authors perform a relocalization based on the seen landmarks. With

(a) Robot equipped with a LiDAR. [121]



(b) Terrain model. [121]

Figure 3.5: The terrain model is used to contract the feasible set of poses of the robot. The images are adapted from [121].

this procedure, the authors report a lower localization error than a classical EKF-localization approach, which is less robust concerning outliers.

The main problems interval-based localization approaches have are pessimism and rigid outlier treatment. One way to deal with the pessimism is to apply a SIVIA approach that performs more bisections. Unfortunately, this leads to a higher computational burden. Hence, a trade-off between pessimism and computational effort is necessary. Regarding outliers, all presented methods introduce q-relaxed intersection, which can identify at most q outliers. The main problem of this robustification approach is that we need to know a priori the maximum number of outliers. However, this cannot be known beforehand. That means selecting q is not trivial and has a tremendous impact on the solution: If q is chosen too small, outliers will be considered in the solution, and the estimated solution set is not guaranteed anymore – or even becomes empty. Good and restrictive measurements may be omitted if q is chosen too high, and the estimation can become overly pessimistic. Due to this dilemma, Jaulin introduces the GOMNE algorithm [49, 121] that selects the minimal q for which the solution set is not empty. However, this algorithm leads to two problems. First, the selected q does not guarantee that all outliers can be detected. Hence, the solution set can still be corrupted by outliers. Second, the iterative nature of GOMNE makes the approach costly in computation.

3.3 Hybrid Approaches

Hanebeck and Horn introduce in [123] a mixed stochastic and set-theoretic uncertainty model. The authors describe a measurement error composed of two additive parts. On the one hand, stochastic noise cannot be bounded, but its probability density is well described in most cases by a normal distribution. On the other hand, measurements can be further corrupted by systematic biases for which rigid bounds can be defined, but the distribution is unknown. By defining the error by two additive terms, the authors derive an estimator containing classical probabilistic and set-theoretic estimation concepts as border cases. The authors show that this approach can be applied to localization problems in a simulation. On the one hand, the more measurements of one state are acquired, the smaller the stochastic uncertainty term will get.

Nonetheless, the set-theoretic uncertainty does not depend on the number of measurements. It does not shrink, due to which the authors show that the estimation result always includes the correct solution. However, the authors only present a simple scalar localization problem along one axis.

Jaulin presents in [124] a probabilistic approach that uses classical set-membership localization methods for state estimation. The method's main idea is to provide a lower bound for the probability that the robot is located in the set estimated by classical set-membership localization. Hence, instead of defining the different uncertainties as an additive term as suggested in [123], Jaulin proposes to provide a quality measure for the interval-based estimate with probabilistic means. Therefore, Jaulin suggests exploiting the probability of the occurrence of an outlier. The author shows, under the assumption that the occurrence of an outlier at a specific timestep is independent of the past, that the probability of having exactly i inliers among m follows the binomial law. Hence, this approach is compatible with the fact that although a q-relaxed set-membership-based localization provides a feasible set of poses, there is still a non-zero probability that the robot is not inside the estimated set. This is because the occurrence of outliers is seen probabilistically, and accordingly, more than q outliers might occur – but with a lower probability. As a result, Jaulin provides an interpretation of the q-relaxed set-membership-based localization estimate in the context of probabilities and shows that both strategies of modeling the error are compatible.

Nassreddine et al. [17] propose a hybrid state estimation approach that applies the Dempster-Shafer theory. The core idea is to extend the interval representation of a set by subsets with mass-functions that can be interpreted as probabilities similar to [124]. Hence, partial information on the distribution of the measurements can be taken into account. Although the authors demonstrate that the approach leads to less pessimistic results than pure interval-based approaches, the proposed method is costly in computation since maintaining the mass-functions for the subsets generates a computational overhead.

Ashokaraja at al. [125] present a hybrid localization approach that fuses the EKF with an interval-based localization approach. The authors equipped the mobile robot with inertial sensors, encoders, and ultrasonic range sensors. Since the EKF is only used to estimate the vehicle's pose using the inertial sensors data and the encoder data, the estimate provided by the EKF suffers from biases and drift. The interval approach is used to correct this accumulating error: In the interval approach, the authors employ a line segments map of the environment and the ultrasonic range measurements as suggested in [109]. As a result, the interval approach provides a set of feasible poses on the map, considering the local measurements. In contrast, the EKF provides a pose estimate based on inertial and wheel encoder data. Two cases can occur to perform the adaptive fusion mechanism of both estimates. If the interval-based estimate encloses the EKF-based pose estimate, the EKF pose is qualified as valid. However, if the EKF pose estimate is not inside the feasible set, a pose inside the feasible set is selected that is geometrically closest to the invalid EKF estimate. Hence, the EKF estimate gets corrected. The authors experimentally show that the adaptive fusion mechanism of both approaches leads to lower estimation errors than the EKF-only estimation. Additionally, the authors not only provide a feasible set, but this fusion mechanism also provides a point estimate of the pose that can, for instance, be used for control algorithms. Nonetheless, the proposed approach does not exploit the full potential of the individual methods: The EKF can take the

ultrasonic measurements for the localization into account, which will reduce the drift. Also, the interval-based method can be improved by including inertial and encoder data in the pose estimation relating consecutive poses to each other, thereby reducing the pessimism.

Louédec and Jaulin [126] also propose to combine an interval-based filter with the EKF. However, instead of applying the algorithms separately on each set of sensors and fusing the estimate at a later point in the state estimation, the core idea presented in [126] is to use the interval-based filter to narrow down the feasible set that provides potential points for the linearization. In this approach, the interval method helps the EKF to find a proper linearization point. As a result, the proposed Interval EKF (IEKF) will behave as a classical EKF as soon as the state estimate is close to the correct solution. In contrast, when the estimate is far from the correct solution, the IEKF will benefit from the global view of the set-membership approach. The authors apply this approach to an underwater robot localization that can sense its distance to beacons. The authors' experimental results show that the EKF diverges if the initial estimate is too far from the correct solution. By using the interval approach to provide the EKF with the proper linearization points, the IEKF can converge to the correct solution independent of the initial pose estimate.

Abdallah et al. [127] present a hybrid interval-probabilistic state estimation approach called *Box-Particle Filter* (BPF). The main idea is to replace punctual states in particle filters with boxes. Hence, instead of spreading point-valued particles, boxes are spread. This makes it possible to significantly reduce the number of particles – since a box represents an infinite set of point-valued particles – and to increase computational efficiency. To accomplish the BPF, the processing steps of the particle filter need to be adapted to interval-based computations. The prediction step can easily be adopted by applying inclusion functions to the propagation equations. However, the weighting cannot be easily adapted to the boxes. While in the classical particle filter the weight is determined based on the proximity between the real and the predicted measurements given the predicted state, the BPF takes the *box likelihood* as a measure of the weight. Therefore, the intersection between predicted and real measurements is determined. The authors define the box likelihood as the ratio between the interval widths of the intersected and predicted measurements. Hence, the larger the overlap between the intersected and predicted measurement is, the higher the weight for this particular measurement. As multiple measurements are associated with one box-particle, the product among all ratios is defined as the box likelihood of the box-particle. Before resampling, the authors propose contracting the box-particles to eliminate inconsistent parts. The same strategies used in the classical particle filter can be applied to the weighted random resampling. The authors apply the BPF to localization problems using GPS, gyro, and odometer data. While the gyro and odometry data is used for state prediction, the GPS data is used for weighting. Wang et al. extend in [128] the BPF to line features to perform localization with OSM building footprints. The BPF is computationally more efficient than the classical particle filter but is more pessimistic in the state estimation. The BPF also suffers from the particle depletion problem as the classical particle filter due to the random resampling. Hence, the BPF may also suffer from box-particle depletion.

In contrast to [127], Neuland et al. [11, 95, 129] propose to fuse the particle filter and the set-membership approaches for localization tasks differently. While Abdallah et al. spread boxes instead of particles, Neuland et al. spreads particles inside the feasible set of poses determined

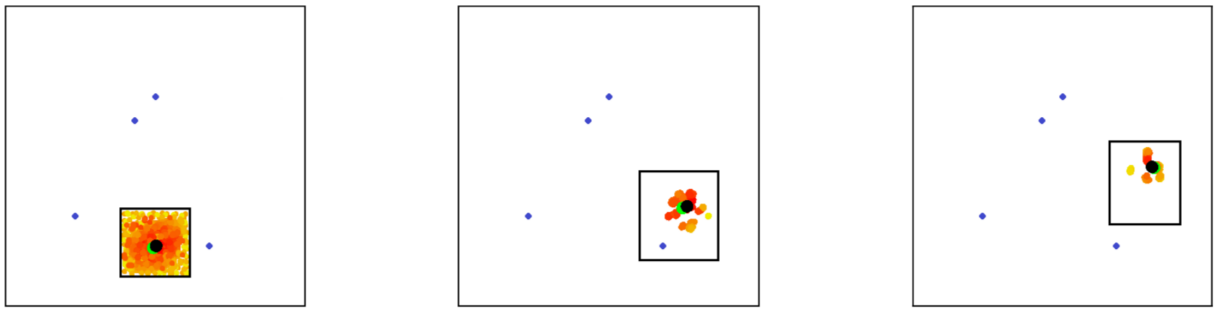


Figure 3.6: Evolution of the hybrid method using contractors that reduce the feasible set combined with MCL. The black boxes represent the feasible set. The particles are colored yellow to red depending on the weight, the blue dots are landmarks, and the green dot is the average particle. [11]

by the set-membership approach. As a result, the authors propose a localization scheme that consists of two steps. First, the initial pose estimate, which can be arbitrarily large, is narrowed down to a feasible set that is consistent with the local measurements. Second, particles are spread inside the feasible set, and the particle filter algorithm can be applied to refine the pose estimate. Consequently, the interval-based approach helps the particle filter stick to the feasible region, as shown in Figure 3.6. Therefore, in the first iteration of the particle filter, the initial population of particles is only created inside the search space defined by the feasible set. The particles are evaluated and discarded for subsequent iterations if they are located outside the feasible set. For each discarded particle, a random particle is drawn inside the feasible set and is added to the current population. The authors show that this approach leads to better coverage of the uncertainty region since the interval-based method reduces the search space. Further, wrong convergence can be detected as the interval approach guides the particle filter to stick to the feasible solutions. The authors evaluate their approach only with simulated data. While in [11], the authors extend the contractor-only approach for the interval-based method to a SIVIA approach that provides tighter bounds on the feasible set, in [95], the authors further robustify the interval approach with a q-relaxed version of SIVIA (RSIVIA). Further, in [95], the authors introduce the global localization problem in simulation. However, experiments with a real robot are not yet conducted with the proposed approach. Additionally, the authors report high runtimes for the calculations, which makes their implementation inapplicable to real-time applications.

4 Visual Odometry

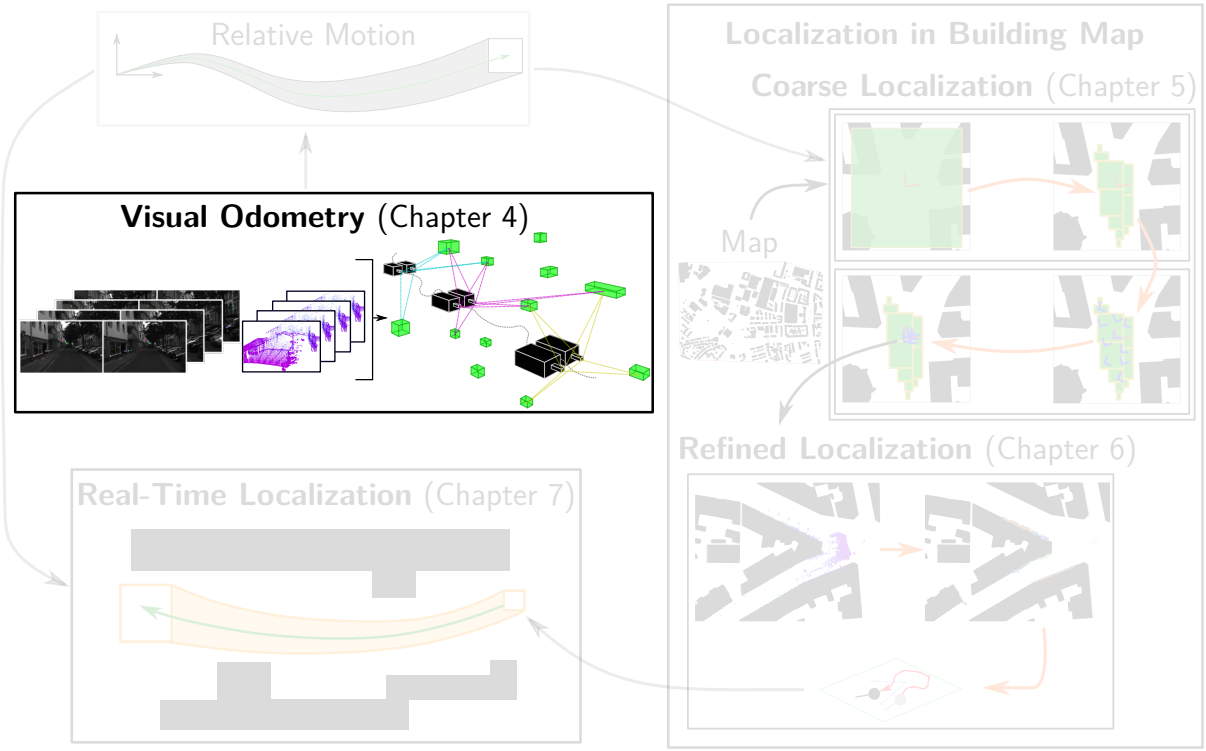


Figure 4.1: The visual odometry is the first module in the localization pipeline.

The first module in our HyPaSCoRe Localization pipeline is the visual odometry, as highlighted in the overview in Figure 4.1. The main goal of this chapter is to determine the vehicle's relative motion. Note that this module does not require prior maps since it solely uses local sensor data to compute the relative movement. The close-up on the visual odometry module is presented in Figure 4.2. As input data, this module uses stereo images and LiDAR data. Based on the input data, visual odometry is performed by constructing a SLAM-graph and solving the graph problem by applying classical least squares optimization and interval analysis in a hybrid fashion. A SLAM-graph is illustrated on the right side of Figure 4.2. The trajectory is represented by a sequence of pose nodes, visualized by black boxes, and landmarks are represented by landmark nodes, visualized by green boxes. The connecting edges visualize the observation constraints between the poses and landmarks.

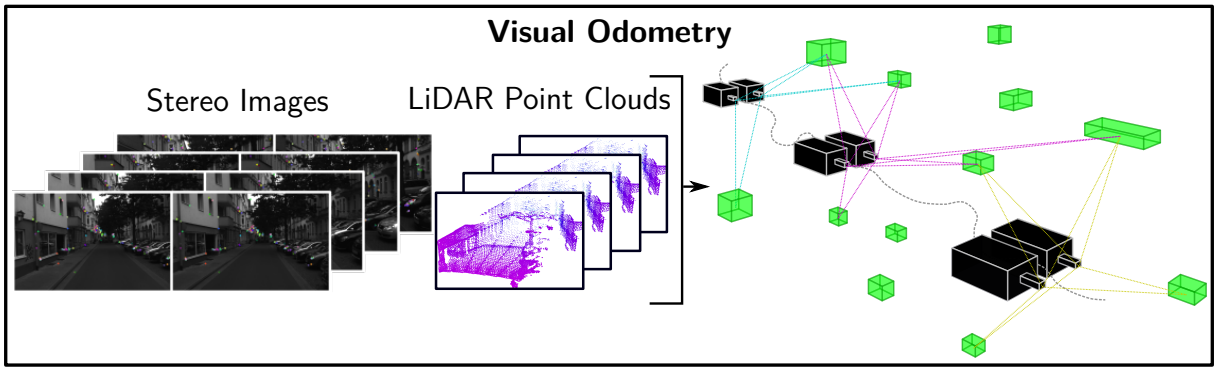


Figure 4.2: Visual odometry. Stereo images and LiDAR data are used to determine the relative motion of the vehicle. Therefore, a SLAM-graph is constructed and solved.

This chapter is structured as follows: First, we will introduce the assumptions and notations in the hybrid visual odometry approach in Section 4.1. Our visual odometry module consists of two parts. The first part is the construction of the SLAM-graph based on the incoming data stream introduced in Section 4.2. The core idea of constructing the SLAM-graph is to structure the measurements to represent the links and dependencies between the involved variables. This part, also called the front-end, performs its computations in real-time, directly evaluating and inserting the local sensor data into the graph. The second part is evaluating the graph, which we call the back-end. The back-end process runs parallel to the front-end but with a lower processing frequency. The goal in the back-end is to estimate the vehicle poses and the landmark locations. The central contribution of this chapter is the preselection of consistent landmarks in the back-end for the interval-based visual odometry computation employing a probabilistic windowed bundle adjustment. The back-end is presented in Section 4.3. We will conclude the chapter by putting the visual odometry module in the HyPaSCoRe Localization context in Section 4.4.

4.1 Task Description, Notation and Assumptions

The goal of this chapter is to determine the relative motion of the vehicle. The literature review in Appendix B reveals that visual odometry methods that simultaneously build a local map for pose estimation provide the most accurate results. We will stick to such an approach. Nonetheless, as we are only interested in the vehicle's relative motion, we do not incorporate any loop closure refinements.

In the scope of this chapter, we use a stereo camera system and a LiDAR scanner. We assume the stereo camera is calibrated, and the system provides stereo-rectified images. Furthermore, we assume that all extrinsic calibration parameters are precisely known. The sensors provide synchronous data. We also assume that the LiDAR data is motion-compensated.

Figure 4.3 shows the data stream we assume from the sensors on a timeline. Both images and LiDAR scan are synchronously available at a specific time step t with a specified data rate. We define a frame $\mathcal{F}_t$ as a collection of data synchronously acquired at time t. To each frame $\mathcal{F}_t$, we associate a pose that we represent in the SLAM-graph by a pose node. An exemplary

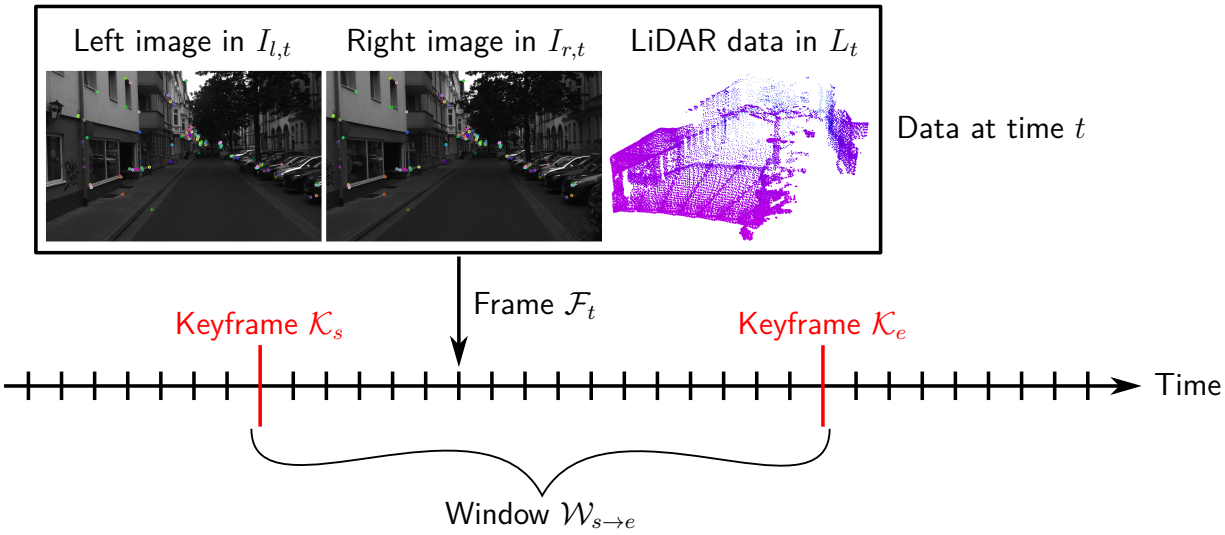


Figure 4.3: The data stream, frames, keyframes, and windows on a timeline. A window $\mathcal{W}_{s\to e}$ is always defined as the set of frames between the times s and e for which keyframes $\mathcal{K}_s$ and $\mathcal{K}_e$ are inserted. A frame $\mathcal{F}_t$ describes the synchronized data collection acquired at time t.

SLAM-graph is shown in Figure 4.4. Pose nodes are illustrated in Figure 4.4 by circular nodes with vehicle symbols.

We obtain landmarks by detecting image features in the stereo images and by reconstructing the 3D location of the feature incorporating the LiDAR depth information. Hence, we define an observation as the pixel points in a frame's left and right image, while a landmark is the corresponding 3D location.

We define a pixel location in the left image of the frame $\mathcal{F}_t$ by ${}^{I_{l,t}}\mathbf{p}$. While the bold $\mathbf{p}$ denotes that the variable represents a general location, the top left index $I_{l,t}$ defines in which coordinate frame this location is described. In this case, the coordinate frame is an image coordinate frame I that spans the 2D image space. The indices l and t encode that the left image corresponding to $\mathcal{F}_t$ at time t is considered. Hence, if we want to describe the location of a pixel in the right image at time t, we have to exchange the top left index by $I_{r,t}$.

Similarly, landmark locations are defined by ${}^{C_{l,t}}\mathbf{p}_i$. The top left index indicates that the location is defined in the coordinate frame $C_{l,t}$, which is the left camera frame at time t. The left camera frame is a 3D coordinate frame, and the location has three parameters. If we want to describe a point in the LiDAR frame at time t, we will exchange the top left index by L_t. Additional index notations can be found in the symbol list.

Landmarks are also represented by nodes in the SLAM-graph. In Figure 4.4, landmark nodes are visualized by stars. Pose nodes and landmark nodes are connected via observation edges. In our case, those edges encode the information on the pixel location in the images of the corresponding frame. To determine the vehicle's relative motion, observing the same landmarks across multiple frames is vital. Therefore, we track features in the images to obtain landmark associations across multiple frames. We do not detect new landmarks in each frame $\mathcal{F}_t$. However, if we enter new scenes in the environment, we will have to detect new landmarks that we can track subsequently, as explained in Subsection 4.2.1. Therefore we introduce so-called keyframes $\mathcal{K}_t$, which are special frames at which we detect new landmarks. Note that for normal

frames $\mathcal{F}_t$, landmarks are only tracked and never newly detected. As a consequence, on $\mathcal{F}_t$, only landmarks are tracked that were detected in the previous keyframe $\mathcal{K}_s$ that was inserted at time $s < t$. Another consequence of this architecture is, that all frames $\mathcal{F}_t$ with $t \in \{s, ..., e\}$ for $s, e \in \mathbb{N}$ and $s < e$ between two keyframes $\mathcal{K}_s$ and $\mathcal{K}_e$ always track the same landmarks. In the following, we will call this connected sequence of frames $\mathcal{F}_t$ with $t \in \{s, ..., e\}$ that starts and ends with keyframes as a window $\mathcal{W}_{s \to e}$. In the bundle adjustment community, the window is also called a batch. We will use both terms interchangeably. Figure 4.3 visualizes the frames, keyframes, and a window on the timeline. In the exemplary SLAM-graph in Figure 4.4, we visualize the windows by coloring the corresponding nodes accordingly.

The presented approach represents the vehicle's pose by intervals. As we only determine the relative motion of the vehicle, we only store for each frame $\mathcal{F}_t$ the relative motion ${}^{C_{l,t-1}}\mathbf{T}_{C_{l,t}}$, which describes the transformation between the previous frame $\mathcal{F}_{t-1}$ and the currently considered frame $\mathcal{F}_t$ described in the left camera frame rigidly mounted on the vehicle. Internally, we represent the relative transformation by intervals. That means the translation $[{}^{C_{l,t-1}}\mathbf{t}_{C_{l,t}}]$ consists of three intervals. The rotation described by Euler angles in the RPY-convention $[{}^{C_{l,t-1}}\boldsymbol{\xi}_{C_{l,t}}]$ also consists of three intervals. The goal is to determine the set of six intervals by propagating the observation uncertainty of the landmarks in the stereo images augmented by the LiDAR depth information.

4.2 Front-End – Build the SLAM-Graph

The objective of the front-end is to construct the SLAM-graph. We use the SLAM-graph as a data structure that conveniently stores the dependencies and links between the variables. In our case, we have two types of variable sets: pose variables and landmark variables. Since we use natural visual features as landmarks, we first need to detect distinct and well track-able features in the stereo-camera system as described in Subsection 4.2.1. After that, using the tracked features in the left and right image, we perform a stereo reconstruction by extending the epipolar geometry to interval analysis (see Subsection 4.2.2). As a result, we obtain boxes describing the outer bound of the seen feature. To reduce the uncertainty of the feature boxes, we further incorporate the LiDAR-data as described in Subsection 4.2.3 that provides accurate range data. As we track visual features frame by frame, we also compute in real-time the dead-reckoning as explained in Subsection 4.2.4 to obtain an initialization for the windowed bundle adjustment described in the next section. Note that all processing steps are performed in real-time. An exemplary SLAM-graph is illustrated in Figure 4.4.

4.2.1 Image Feature Tracking

Initially, no features are tracked when we start the visual odometry pipeline. Hence, the very first frame has to be defined as a keyframe, and we need to detect new features. As described in the State of the Art summary in Chapter B, the computer vision community provides many different types of features. However, our application needs features that can be detected and compared as fast as possible with less computational effort. Therefore, we have decided to use Oriented FAST and Rotated BRIEF (ORB) features [130], which have proven to be a good

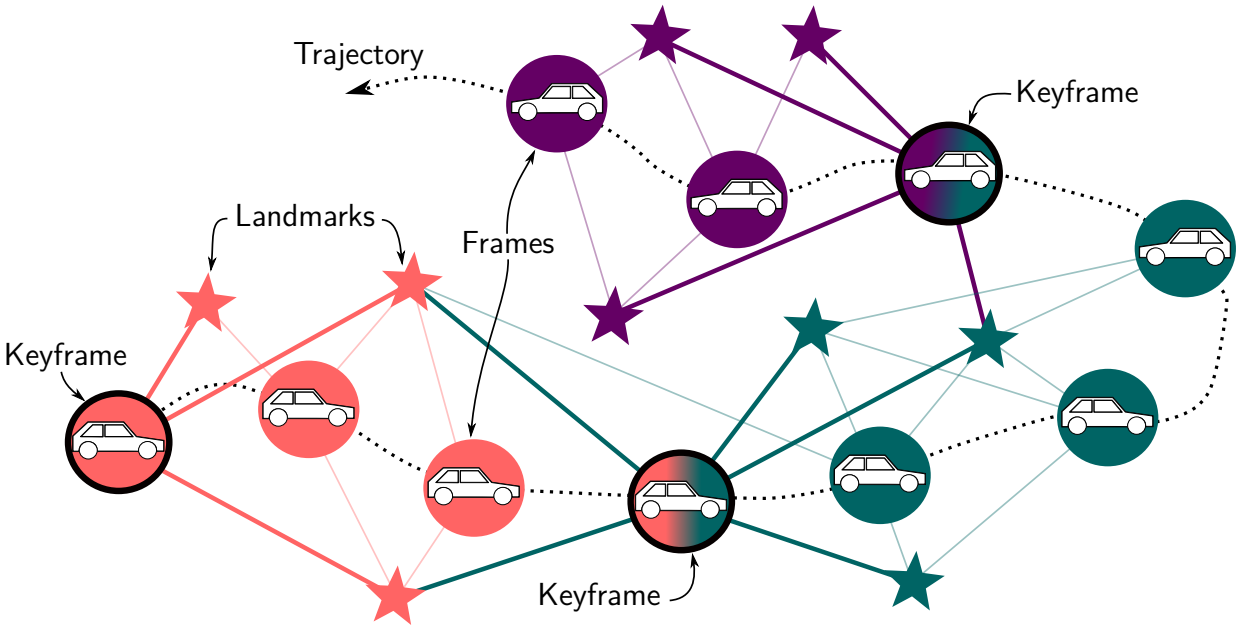


Figure 4.4: SLAM-Graph. The vehicle poses at different points in time are represented by circular pose nodes along the dotted trajectory that we seek to determine. Pose nodes correspond to frames. Keyframes are highlighted with a black border. Landmarks are represented by star-shaped nodes connected to the pose nodes via observation edges. Nodes that belong to a specific window are accordingly color coded. In total, three windows are illustrated.

choice according to [26, 131–134] for relative motion estimation. Since we use a calibrated stereo-camera system that provides two stereo-rectified images, we initially detect ORB-features in the left and right images independently. A good distribution of the features in the image is vital for well-constrained pose estimation. Therefore, we perform feature detection in predefined tiles of the image as proposed in [134]. As shown in Figure 4.5, the image plane is divided into multiple tiles, and the ORB-feature detection is performed on each.

Our goal is to detect features commonly observed in the left and right images to triangulate the feature location. Therefore, we match the left and right features. Since we have stereo-rectified images, the epipolar lines in the images are horizontal (see Part 2.3.1.2). As a result, corresponding features in the left and right images need to be located on the same image row. However, due to observation uncertainties and imperfect calibration, the detected features may not perfectly meet the epipolar constraint. Hence, we consider an uncertainty region above and below the image row. Depending on the feature's detection scale, we adapt the uncertainty region that defines a stripe in which we search for a given feature. Note that we consider the left image features to query in the right image for features located inside the search region defined by the stripe around the respective image row. We compare each left image feature with all potential matches in the right image within the search region stripe. The feature similarity is determined by the Hamming distance between the binary-oriented BRIEF descriptors [130]. If the match with the lowest Hamming distance is lower than a predefined threshold, the respective left and right features are seen as a correctly matched feature pair. Figure 4.6 shows a frame's left and right images with matched features. Note that each feature can be matched independently, making parallel processing possible. We use OpenMP

Figure 4.5: ORB-feature detection and image tiles. The detected features are colored. The image plane is divided into tiles with identical sizes in which features are detected. White lines visualize the border of the tiles. This enables to detect features that are well distributed on the image plane.

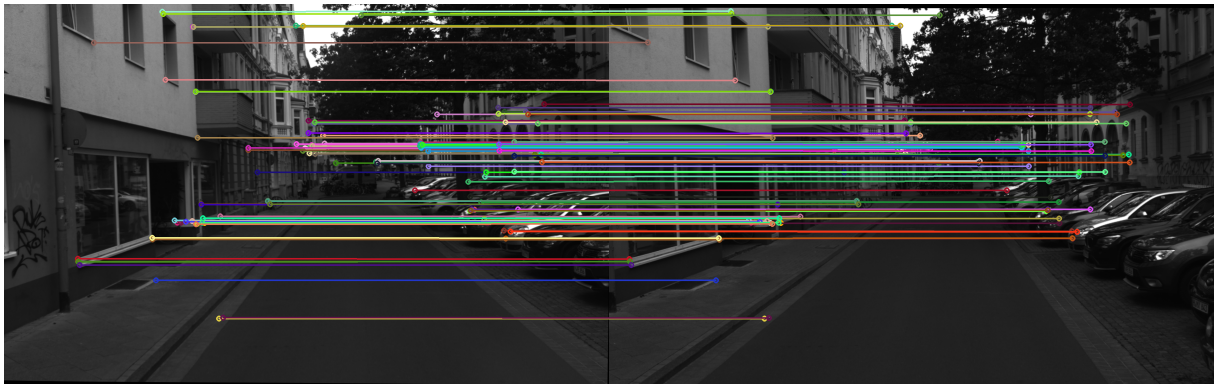


Figure 4.6: Feature matching in left and right images. Detected and tracked ORB-features in the left and right images are matched based on the binary BRIEF descriptor. Matched features are highlighted with the same color, and a connection line indicates their correspondence. Since we use stereo-rectified image pairs, the epipolar lines are horizontal. That is why the corresponding features in the left and right images are on the same image row.

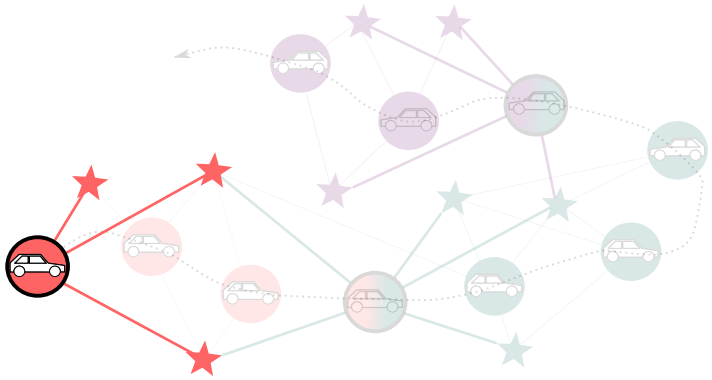


Figure 4.7: Initialization of the SLAM-graph with the first keyframe.

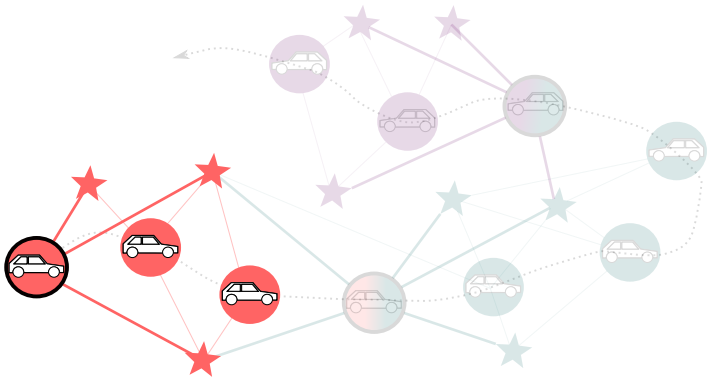


Figure 4.8: Successfully tracked frames are inserted to the SLAM-graph.

[135] for multi-threaded processing. Figure 4.7 illustrates the initialization with the first frame as a keyframe and the observed landmarks in the SLAM-graph.

If the feature tracking was initialized and features were detected in the previous frame, we try to track the features in the images subsequently which is also known as registration problem. To track features, we apply the Lucas-Kanade optical flow algorithm [136] for each feature that was successfully matched in the previous frame $\mathcal{F}_{t-1}$ in the left and right image. The optical flow algorithm provides the pixel error on the tracking result for each feature. We only consider features tracked in both images with an error lower than a defined threshold. Furthermore, if the tracked features in the left and right image satisfy the epipolar constraint and the classically reconstructed feature location (cf. Part 2.3.1.2) has positive depth, we consider the feature to be a correctly tracked landmark observation. As a result, if enough features are tracked, those frames $\mathcal{F}_t$ that were successfully tracked are inserted into the graph as pose nodes connected to the respective landmark nodes as illustrated in Figure 4.8.

However, when the number of tracked features drops – since the vehicle moved, for instance, into parts of the scene that were not captured before or features are not well tracked, new features need to be detected. As defined above, keyframes are those frames in which we need to detect new features. Hence, a keyframe $\mathcal{K}_t$ provides new landmarks and observations while the regular frames $\mathcal{F}_t$ only provide new observations for landmarks that were already detected in the last keyframe $\mathcal{K}_s$ for $s < t$.

If we need to insert a new keyframe, we must prevent re-detecting features in texture-rich image regions. Therefore, we only perform detection on image tiles that contain too few features. The decision in which tile new features should be detected is based on the left image in our implementation. To decide when to insert a new keyframe – that means when we have to detect new landmarks, we exploit the distribution of features in the left image again: For each tile, we check how many features are tracked for a new frame. If a tile has more than a defined threshold for the minimal number of features, the tile is qualified as good. If less than $40\,\%$ of the tiles are good, new features are tracked, and the current frame is defined as a keyframe $\mathcal{K}_t$. Note that we only detect new features in those tiles that were qualified as not good. As a result, very stable landmarks can be tracked for a long trajectory surviving across multiple keyframes. Figure 4.9 illustrates the insertion of a keyframe. As the keyframe introduces the end and start of windows, the keyframe node is colored with both window colors.

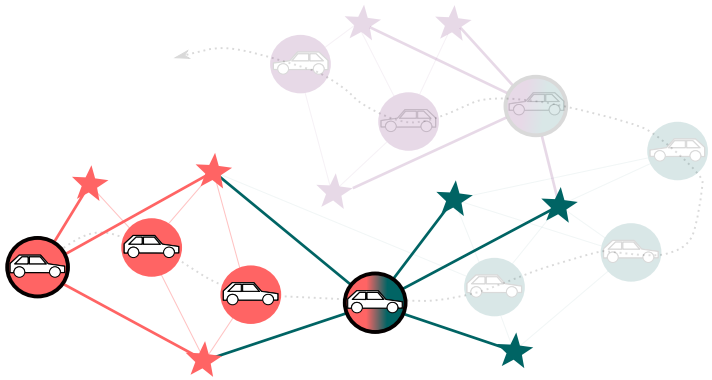


Figure 4.9: Insertion of a keyframe into the SLAM-graph if not enough features are tracked.

The image feature tracking provides the information on which landmarks are seen from the current frame $\mathcal{F}_t$. We encode this information in a SLAM-graph structure that stores for each frame $\mathcal{F}_t$ which landmark i was seen and the corresponding pixel positions ${}^{I_{l,t}}\mathbf{p}_i$ and ${}^{I_{r,t}}\mathbf{p}_i$ of the feature in the left and right image as observations. Furthermore, we compute the 3D location ${}^{C_{l,t}}\mathbf{p}_i$ of the landmark i described in the local left camera frame as an additional component of the observation. The 3D reconstruction is explained in the following.

4.2.2 Interval-based Stereo Reconstruction

We only permit features consistently observed through the left and right cameras. Since the stereo camera system is calibrated, we can apply the epipolar geometry to compute the local feature position in the 3D scene described in the left camera frame. First, we introduce the classical triangulation method that provides the intersection point that minimizes the quadratic distance between the observation rays. After that, we will introduce the interval-based counterpart implementing a feature-box contractor.

Let ${}^{I_{l,t}}\mathbf{p} = \begin{pmatrix} x_l & y_l \end{pmatrix}^T$ be the pixel position of the feature in the left image and ${}^{I_{r,t}}\mathbf{p} = \begin{pmatrix} x_r & y_r \end{pmatrix}^T$ the pixel position in the right image. Furthermore, we define the unknown 3D feature position in the left camera frame as ${}^{C_{l,t}}\mathbf{p} = \begin{pmatrix} X_l & Y_l & Z_l \end{pmatrix}^T$. Since we obtain stereo-rectified images captured by identical cameras, the intrinsic camera matrix $\mathbf{K}$ for both cameras is identical:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.1}$$

The main advantage of using stereo-rectified images is the assumption of images synthetically captured by perfectly parallel cameras that only have a translational displacement in the horizontal direction, as explained in Part 2.3.1.2. As a consequence, the feature location described in the right camera frame is ${}^{C_{r,t}}\mathbf{p} = \begin{pmatrix} X_l - b & Y_l & Z_l \end{pmatrix}^T$, where b is the stereo base-length that measures the distance between the projection points of the left and right

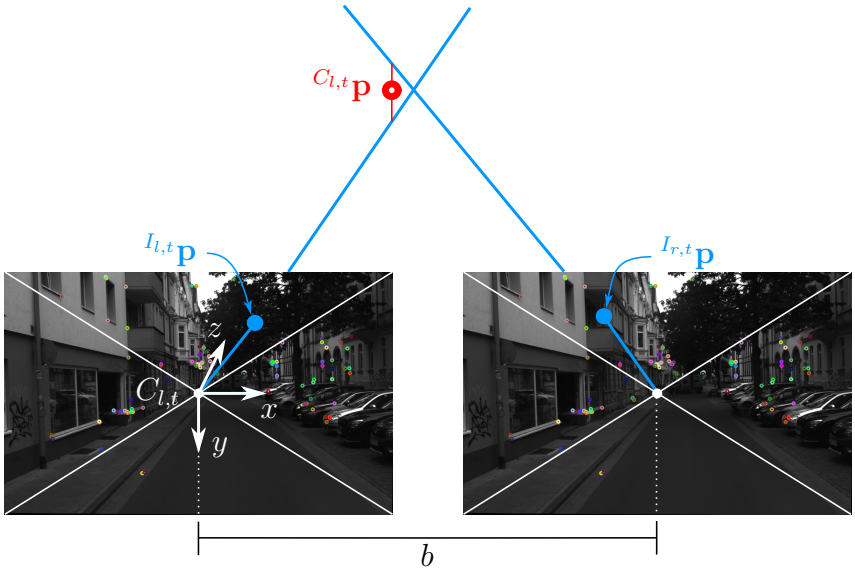


Figure 4.10: Feature location triangulation with least squares approach. The blue points represent matched features $^{I_{l,t}}\mathbf{p}$ and $^{I_{r,t}}\mathbf{p}$ in the images. The projection centers are visualized by white dots in the center of the images. The observation rays are colored blue. In this example, observation rays do not intersect. The least squares approach provides the feature location $^{C_{l,t}}\mathbf{p}$ that minimized the quadratic distance between the observation rays.

camera. As a consequence, we can use the left and right camera projection equation to formulate the linear equation system for triangulating $^{C_{l,t}}\mathbf{p}$

$$\begin{cases} \begin{pmatrix} ^{I_{l,t}}\mathbf{p} \\ 1 \end{pmatrix} \cdot Z_l = \mathbf{K} \cdot {}^{C_{l,t}}\mathbf{p} \\ \begin{pmatrix} ^{I_{r,t}}\mathbf{p} \\ 1 \end{pmatrix} \cdot Z_l = \mathbf{K} \cdot {}^{C_{r,t}}\mathbf{p} \end{cases} . \tag{4.2}$$

The equation system yields six equations. However, the homogeneous extension in K only leads to four equations we can use to solve the system. As a consequence, we can reformulate the equation system in matrix form

$$\begin{pmatrix} -1 & 0 & \frac{x_l - c_x}{f_x} \\ 0 & -1 & \frac{y_l - c_y}{f_y} \\ -1 & 0 & \frac{x_r - c_x}{f_x} \\ 0 & -1 & \frac{y_r - c_y}{f_y} \end{pmatrix} \cdot {}^{C_{l,t}}\mathbf{p} = \begin{pmatrix} 0 \\ 0 \\ -b \\ 0 \end{pmatrix} . \tag{4.3}$$

Note that $^{C_{l,t}}\mathbf{p}$ only has three unknown parameters, while we obtain four equations. Hence, the linear equation system is overdetermined. We solve the equations system in the least squares fashion by applying a column pivoting QR decomposition. Figure 4.10 visualizes the result we obtain by solving (4.3): The reason for the overdetermination of the system is that observation rays may not intersect. That is why we determine the point that minimizes the quadratic distance between the observation rays as indicated in Figure 4.10.

While we use the least squares reconstruction to validate if a feature match is consistent, as explained in Subsection 4.2.1, the more interesting part is the interval feature location that considers the observation uncertainties. Therefore we will design a contractor. Hence, let us again consider (4.3) that provides the constraints on the feature box considering the local image observations. Note that the last column of the left matrix has no zero entries. Since this column corresponds to the depth coordinate Z_l, the depth parameter substantially impacts X_l and Y_l. Furthermore, we can see in (4.3) that only the first and third rows determine Z_l due to the horizontal stereo structure indicated by the $-b$ entry in the right vector. When we extract the first and third equations from (4.3), we obtain

$$\begin{cases} -X_l + \frac{x_l - c_x}{f_x} Z_l = 0 \\ -X_l + \frac{x_r - c_x}{f_x} Z_l = -b \end{cases} . \tag{4.4}$$

By substituting X_l in the first equation with the corresponding term obtained by the second, we get the stereo-depth constraint

$$Z_l = \frac{b \cdot f_x}{x_l - x_r}. \tag{4.5}$$

The equations in (4.4) do not only constrain Z_l but also X_l. To obtain the constraints on Y_l, we must consider the second and fourth lines in (4.3). Similar to the X_l coordinate, we get

$$\begin{cases} -Y_l + \frac{y_l - c_y}{f_y} Z_l = 0 \\ -Y_l + \frac{y_r - c_y}{f_y} Z_l = 0 \end{cases} . \tag{4.6}$$

Now we have all the constraints to build the contractor for a feature box $[^{C_{l,t}}\mathbf{p}]$. The algorithm for our contractor is depicted in Algorithm 4. Note that directly applying a simple forward-backward contractor based on (4.4) and (4.6) yields poor results since the depth coordinate Z_l appears in all equations. However, by first contracting the depth in line 3 and using the depth estimate to apply the forward-backward contraction using (4.4) in lines 4 and 5, and (4.6) in lines 6 and 7 provides good results on the feature box. Furthermore, we consider the rectified stereo constraints, that corresponding features need to be on the same image row, by contracting the y-component of the pixel boxes in lines 1 and 2.

The visual illustration of the feature box contraction is shown in Figure 4.11. While the classical least squares approach yields the point that minimizes the quadratic distance to line-shaped rays, applying interval analysis, the rays inflate to cones since we consider the pixel observation uncertainty. The intersection of those cones is the region where the observed landmark has to be located. Since the intersection region has a complex shape in 3D space, interval analysis provides the tools to compute the axis-aligned box $[^{C_{l,t}}\mathbf{p}]$ that encloses this region.

4.2.3 LiDAR measurements

The main problem of stereo camera-based 3D reconstruction is the significant depth uncertainty for distant landmarks. The depth uncertainty is high if the stereo system's base length is much smaller than the distance of the feature as the parallax becomes small. This phenomenon is

Algorithm 4: Stereo-Feature Contractor

Data: $[^{I_{l,t}}\mathbf{p}] = \begin{pmatrix}[x_l]\\[y_l]\end{pmatrix}$, $[^{I_{r,t}}\mathbf{p}] = \begin{pmatrix}[x_r]\\[y_r]\end{pmatrix}$, $[^{C_{l,t}}\mathbf{p}] = \begin{pmatrix}[X_l]\\[Y_l]\\[Z_l]\end{pmatrix}$, $\mathbf{K}$, b

Result: $[^{C_{l,t}}\mathbf{p}] = \begin{pmatrix}[X_l]\\[Y_l]\\[Z_l]\end{pmatrix}$

// Rectified stereo constraint

1 $[y_l] = [y_l] \cap [y_r]$;

2 $[y_r] = [y_r] \cap [y_l]$;

// Depth constraint for rectified stereo

3 $[Z_l] = [Z_l] \cap [\frac{b \cdot f_x}{[x_l]-[x_r]}]$;

// Constraints on the X_l-coordinate

// Forward

4 $[X_l] = [X_l] \cap [[Z_l] \cdot \frac{[x_l]-c_x}{f_x}] \cap [[Z_l] \cdot \frac{[x_r]-c_x}{f_x} + b]$;

// Backward

5 $[Z_l] = [Z_l] \cap [[X_l] \cdot \frac{f_x}{[x_l]-c_x}] \cap [[[X_l] - b] \cdot \frac{f_x}{[x_r]-c_x}]$;

// Constraints on the Y_l-coordinate

// Forward

6 $[Y_l] = [Y_l] \cap [[Z_l] \cdot \frac{[y_l]-c_y}{f_y}] \cap [[Z_l] \cdot \frac{[y_r]-c_y}{f_y}]$;

// Backward

7 $[Z_l] = [Z_l] \cap [[Y_l] \cdot \frac{f_y}{[y_l]-c_y}] \cap [[Y_l] \cdot \frac{f_y}{[y_r]-c_y}]$

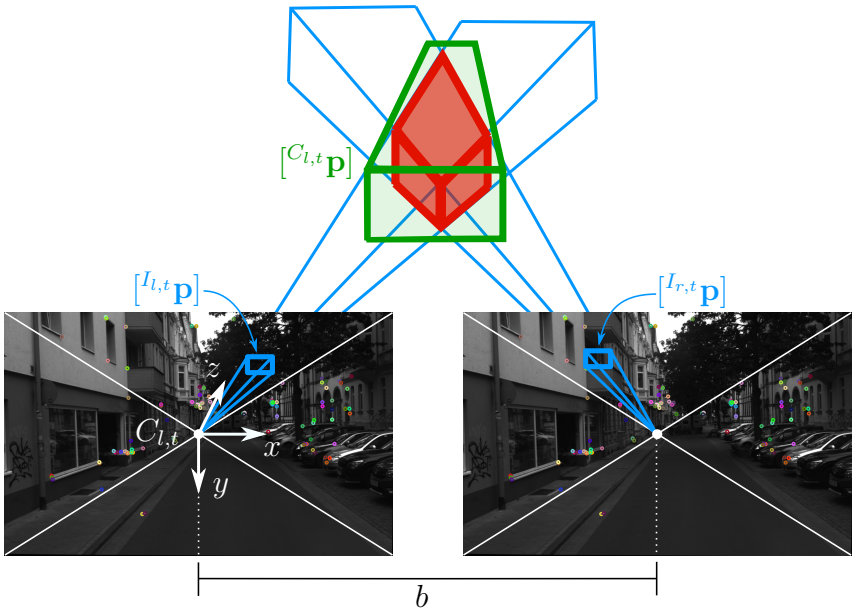


Figure 4.11: Feature location triangulation with interval analysis. The features are represented by the boxes $[^{I_{l,t}}\mathbf{p}]$ and $[^{I_{r,t}}\mathbf{p}]$ in the images. The boxes span cones in 3D space. The intersection of the cones is colored red and describes the region where the tracked feature has to be located. Due to the wrapping effect by applying interval analysis, we obtain the green enclosing box $[^{C_{l,t}}\mathbf{p}]$ by applying the contractor.

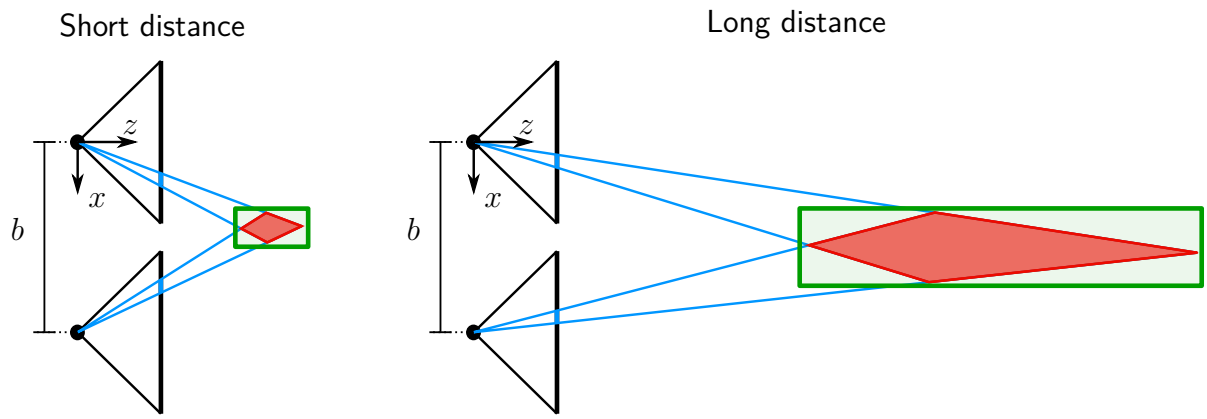


Figure 4.12: Interval-based stereo feature reconstruction uncertainty for different depths. The visualization shows the same reconstruction approach as shown in Figure 4.11 from the top view. On the left, the reconstruction result of a close feature is visualized. On the right, a distant feature is considered that results in a more uncertain reconstruction for the same pixel uncertainty.

illustrated in Figure 4.12. For features located on textured planes like road signs or building facades, the depth measurements can be refined by the LiDAR data as its range measurement is typically more accurate. The set-membership-based fusion of stereo feature observations with LiDAR data was already published in [137] and will be introduced in the following.

Let $[{}^{C_{l,t}}\mathbf{p}]$ be the stereo feature location described in the left camera frame. Furthermore, we assume that the relative transformation ${}^{L}\mathbf{T}_{C_l}$, which transforms a point described in the left camera frame to the LiDAR frame, is known accurately from calibration. The LiDAR provides a set of points ${}^{L_t}\hat{\mathbf{p}}_i$, that inflate to boxes $[{}^{L_t}\hat{\mathbf{p}}_i]$ by considering the interval-based LiDAR uncertainty model introduced in Subsection 2.3.2. First, we need to transform $[{}^{C_{l,t}}\mathbf{p}]$ from the left camera frame $C_{l,t}$ to the LiDAR frame L_t. Therefore, we apply ${}^{L}\mathbf{T}_{C_l}$ which provides $[{}^{L_t}\mathbf{p}]$ using the interval inclusion functions for the transformation. Now, we determine the set of LiDAR measurements $\mathcal{S}$, that have a non-empty intersection with $[{}^{L_t}\mathbf{p}]$ so that

$$\mathcal{S} = \{\, i \in 1, ..., N \mid [{}^{L_t}\mathbf{p}] \cap [{}^{L_t}\hat{\mathbf{p}}_i] \neq \varnothing \,\}. \tag{4.7}$$

If the number of points – that means $|\mathcal{S}|$ – is large enough, we contract $[{}^{L_t}\mathbf{p}]$ to the hull among all intersections of the boxes considered in $\mathcal{S}$ and the initial box $[{}^{L_t}\mathbf{p}]$ by

$$[{}^{L_t}\mathbf{p}] = \bigcup_{i \in \mathcal{S}} ([{}^{L_t}\mathbf{p}] \cap [{}^{L_t}\hat{\mathbf{p}}_i]). \tag{4.8}$$

This step is illustrated in Figure 4.13. While the green box is the initial stereo feature box, the yellow box is the contracted feature location box considering the associated LiDAR measurements colored in blue with red dots. We transform the contracted feature box to the camera frame $C_{l,t}$. Hence, we obtain the contracted feature boxes taking into account stereo camera observations and LiDAR measurements.

To efficiently determine the intersection of $[{}^{L_t}\mathbf{p}]$ with the LiDAR measurements ${}^{L_t}\hat{\mathbf{p}}_i$, we utilize a KD-Tree structure that stores the mids of the LiDAR measurement boxes. To retrieve the LiDAR measurements that can have a potential intersection with $[{}^{L_t}\mathbf{p}]$, we perform a radius

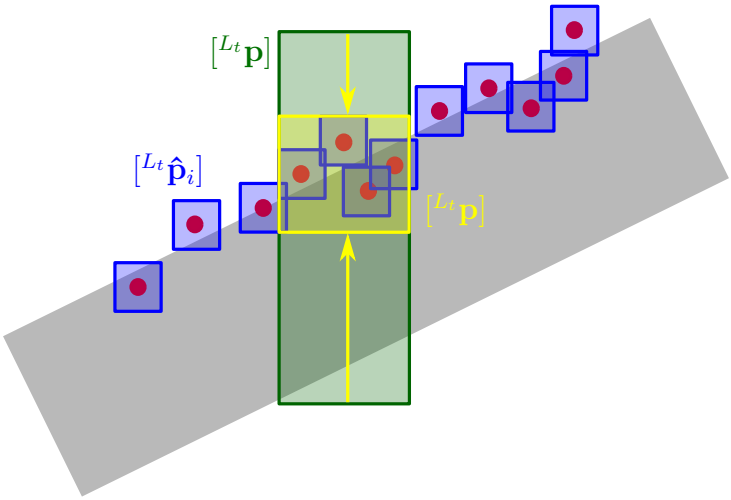


Figure 4.13: Contract stereo feature to the union of intersections with LiDAR measurements. The LiDAR measurements are illustrated by blue boxes with red mid-points. The initial stereo feature box $[^{L_t}\mathbf{p}]$ is visualized green, and the contracted feature box is colored yellow.

search in the KD-Tree using the midpoint of $[^{L_t}\mathbf{p}]$ as search point and the half diameter of $[^{L_t}\mathbf{p}]$ as the radius.

4.2.4 Frame-to-Frame Pose Estimation

Up to this point, we constructed the SLAM-Graph and solved the data association problem. The precise estimation of the pose will be performed in the back-end by applying a windowed bundle adjustment and the interval-based odometry computation, as presented in the next section. However, as we are interested in the vehicle's current pose, we also perform dead-reckoning by determining the current pose of $\mathcal{F}_t$ concerning the previous $\mathcal{F}_{t-1}$. The dead-reckoning result, which only considers a pair of frames, is used to initialize the windowed bundle adjustment in the back-end. Since we only consider the current and previous frame for the dead-reckoning computation, the relative motion transformation is determined with low-weight computation in real-time – well suited for the front-end.

Therefore, we use a classical LM optimization that minimizes the stereo reprojection error as explained in Subsection 2.1.3. The optimization consists of two steps using different robust kernels: First, we apply the Geman-McClure cost function, and second we utilize the saturated robust kernel that clips off all those measurements that exceed a defined error threshold (cf. Subsection 2.1.3). The first optimization step optimizes the pose so that the poses and landmarks fit the measurements as well as possible in the least squares sense while weighting large errors less according to the kernel function presented in [25]. After this optimization step, the MLE result will yield a smaller overall reprojection error. However, it may be corrupted by incorrect matchings (outliers) with large reprojection errors. To reduce the impact of outliers on the MLE result, the optimization with the saturated optimization kernel is applied, which only considers features with an acceptable initial reprojection error. In our implementation, we apply the graph optimization framework g^{2}o [29].

Note that in the front-end, we do not perform an interval-based computation of the relative motion since features that may be qualified as good according to the frame-to-frame optimization may still be corrupted by dynamics: For example, features on slowly moving objects will generate small residuals so that they may be determined as good features. However, since they are outliers, they may incorrectly corrupt the contraction for interval-based dead-reckoning computation. Although there are ways to cope with the outliers using interval analysis (see Subsection 2.2.5) with q-relaxed intersection, the main problem of this approach is that we do not know the number q of inconsistent features. Using the GOMNE algorithm [138] is also not possible here since this may also lead to slightly inconsistent features so that they do not generate empty sets. However, the dead-reckoning results may still be corrupted. Furthermore, GOMNE also leads to longer run times due to its sequential algorithmic structure. That is why we shift the interval-based visual odometry computation to the back-end. There, before applying the interval-based visual odometry computation, we first perform a windowed bundle adjustment that preselects those features that are qualified as good (small residuals for the reprojection in the left and right image) for a longer sequence, where dynamic features are easily detected and can be filtered as inconsistent. Hence, by only taking those features into account that are consistent and lead to lower reprojection errors, we can perform interval-based visual odometry without the q-relaxed intersection since the windowed bundle adjustment already filters all outliers.

4.3 Back-End – Solve the SLAM-Graph

The construction of the SLAM-graph, which represents the data association, was explained in the previous section. An exemplary SLAM-graph is depicted in Figure 4.4. Although the majority of the associations are correct in practice, some may be error-prone: Mismatches of closely located landmarks and features on dynamic objects are possible outliers that may corrupt the interval-based odometry computation. Using the q-relaxed intersection for outlier treatment is not possible in our case since we do not know the number of outliers q beforehand as mentioned above. To identify the outliers and to get rid of those, we utilize prior to the interval-based odometry estimation a robust windowed bundle adjustment. Robust kernels in optimization methods have proven to provide good results, as shown in [26, 27, 29, 101]. Hence, we will use the windowed bundle adjustment augmented by robust kernels to preselect those landmark observations that fall below a defined residual for the reprojection error. We introduce the windowed bundle adjustment in Subsection 4.3.1. The interval-based odometry estimation provides the vehicle's relative motion considering the preselected observations of the landmarks and their uncertainties, and will be presented in Subsection 4.3.2.

The back-end runs on a separate thread parallel to the real-time front-end, similar to the PTAM [139], ORB-SLAM [134], and LSD-SLAM [27] architectures. While the front-end process is always triggered when a new data frame $\mathcal{F}_t$ comes from the sensors, we trigger the back-end process if a new keyframe is inserted that closes the current window of frames and opens a new one. To understand why we consider keyframe insertions to trigger the back-end, we need to analyze Figure 4.3 again. As introduced in Section 4.1, the window size is determined by keyframes: Between two consecutive keyframes $\mathcal{K}_s$ and $\mathcal{K}_e$ with $s < e$ always

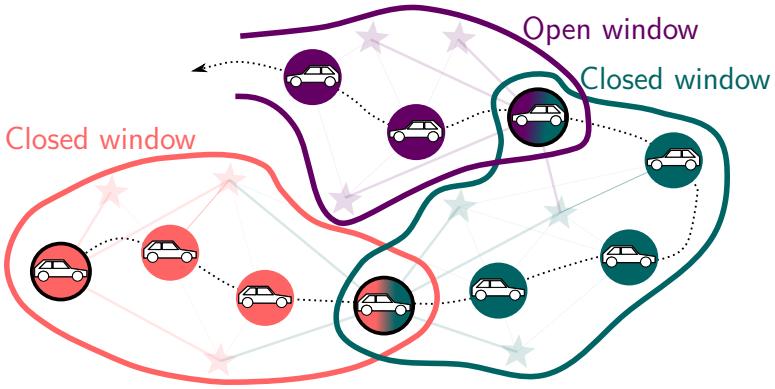


Figure 4.14: In the SLAM-graph, two closed windows and one open window are illustrated. While the windowed bundle adjustment can be performed on the rose and green-colored windows, the violet window must be completed before the optimization can be applied.

the same features are tracked that were detected in the first keyframe $\mathcal{K}_s$ at the beginning of the batch $\mathcal{W}_{s\to e}$. Hence, if a new keyframe $\mathcal{K}_e$ is inserted, the current window of frames with common landmark observations is closed. As a result, we can proceed with the bundle adjustment for this window, as it will not be changed anymore. After the preselection of consistent observations, we compute the interval-based visual odometry for the whole batch. Figure 4.14 illustrates the batches in the exemplary SLAM-graph from Figure 4.4.

The back-end does not need to process the data at frame rate because, in the back-end, a whole window is processed at once. Hence, if a window has a size of 10 at a sensor frame rate of $0.1\,\text{Hz}$, the window encloses a period of $1\,\text{s}$. Within this time, the last window could be processed. Hence, slower processing frequency is possible in the back-end maintaining real-time operability for the visual odometry as we process multiple frames simultaneously. However, the processing must be fast enough to finish the computations in the current window time.

4.3.1 Windowed Bundle Adjustment

The reason for using a windowed bundle adjustment in our ego-motion estimation approach is to detect inconsistent features that do not comply with the largest consistent set of tracked features in the stereo images. As in the major part of the stereo images, static parts of the scene are observed, we can assume that the largest consistent set is the set of static features in which we are interested. This assumption can be violated if a large dynamic object occludes the static scene. To deal with such corner cases, motion sensors like IMUs or wheel odometers need to be fused to the ego-motion estimation. Nonetheless, in the scope of this work, we want to stick to the former assumption of the static scene ignoring the mentioned corner case.

While gross outliers can be detected by checking feature consistency in simple dead reckoning approaches that only compute the relative motion between two consecutive frames [140], small outliers will be hard to detect since they may seem consistent with the correct features. Hence, to cope with such hard-to-detect outliers in the data association, we apply the bundle adjustment approach on a larger window that contains multiple frames. The advantage of a larger observation window is that feature observations with small perturbations, like features

on slowly moving objects, can be detected and discarded from the estimation by applying a robust approach. Figure 4.15 shows features on a slowly moving car that are hard to identify as outliers if we only consider a pair of consecutive frames as suggested in [140]. Nevertheless, when we consider the whole observation window, also small movements will lead to significant reprojection errors for such features in the bundle adjustment, as the slow motion accumulates along the window time. Hence, the windowed bundle adjustment makes detecting the consistent set of landmark observations possible without the need to forecast the number of inconsistent landmarks, as would be the case for the interval counterpart – the q-relaxed intersection – that we avoid with this approach.

We use the same graph optimization framework g^{2}o [29] for the windowed bundle adjustment as for the frame-to-frame pose estimation in Subsection 4.2.4. While we insert the set of frames as pose nodes into the optimization graph, landmarks are inserted as landmark nodes. Observations in the stereo images are the pixel locations of the features corresponding to the respective landmarks represented as edges. Figure 4.4 shows an exemplary graph. The optimization goal is to obtain the poses and the landmark locations so that the observations are met as well as possible, minimizing the quadratic reprojection error in the left and right images.

Since we already structured the captured data in a SLAM-graph, we can directly transfer this SLAM-graph to the g^{2}o optimization graph and perform the optimization. However, a simple least squares bundle adjustment without explicit outlier treatment is doomed to fail, as already a few outliers are sufficient to let the optimization (we use LM) diverge. Therefore, similar to the frame-to-frame pose estimation in Subsection 4.2.4, we perform a two-staged optimization with two different robust kernels. The first optimization step is performed with the Geman-McClure cost function [25] that reduces the impact of a measurement on the optimization the higher the residual is. We initialize the pose nodes with the frame-to-frame pose estimates computed in the front-end. The reference frame for the whole windowed bundle adjustment is the first frame $\mathcal{K}_s$ of the batch, which is always a keyframe as illustrated in Figure 4.4. Hence, first, we transform all poses to the left camera frame $C_{l,s}$ of $\mathcal{K}_s$ by accumulating the frame-to-frame estimations. To initialize the landmark nodes, we need to consider the stereo reconstructed landmark locations seen in $\mathcal{K}_s$: Since the keyframe defines the frame in which new features are detected, it will always have common observations with all following frames $\mathcal{F}_t$ in the window. Hence, we can initialize the landmarks for the whole batch $\mathcal{W}_{s\to e}$ with the stereo reconstructed locations in $C_{l,s}$, while the pose corresponding to $\mathcal{K}_s$ defines the origin of the coordinate system in which we perform the windowed bundle adjustment. The first step provides the initial configuration for the second stage of the optimization. In this stage, we utilize the saturated cost function that discards all those measurements that exceed a defined reprojection error threshold. Hence, this optimization step refines the poses and landmark locations by only considering observations with comparatively low error.

After the two-stage optimization, we can retrieve the remaining reprojection errors for each observation we inserted as an edge to the optimization graph. Now we can proceed with the preselection of the landmarks and the observations for the interval-based visual odometry. For each landmark, we check the corresponding reprojection errors of each observation. Hence, as illustrated in Figure 4.16, we check for each landmark the connected edges – that means observations – and evaluate the reprojection error after the optimization. Suppose there is

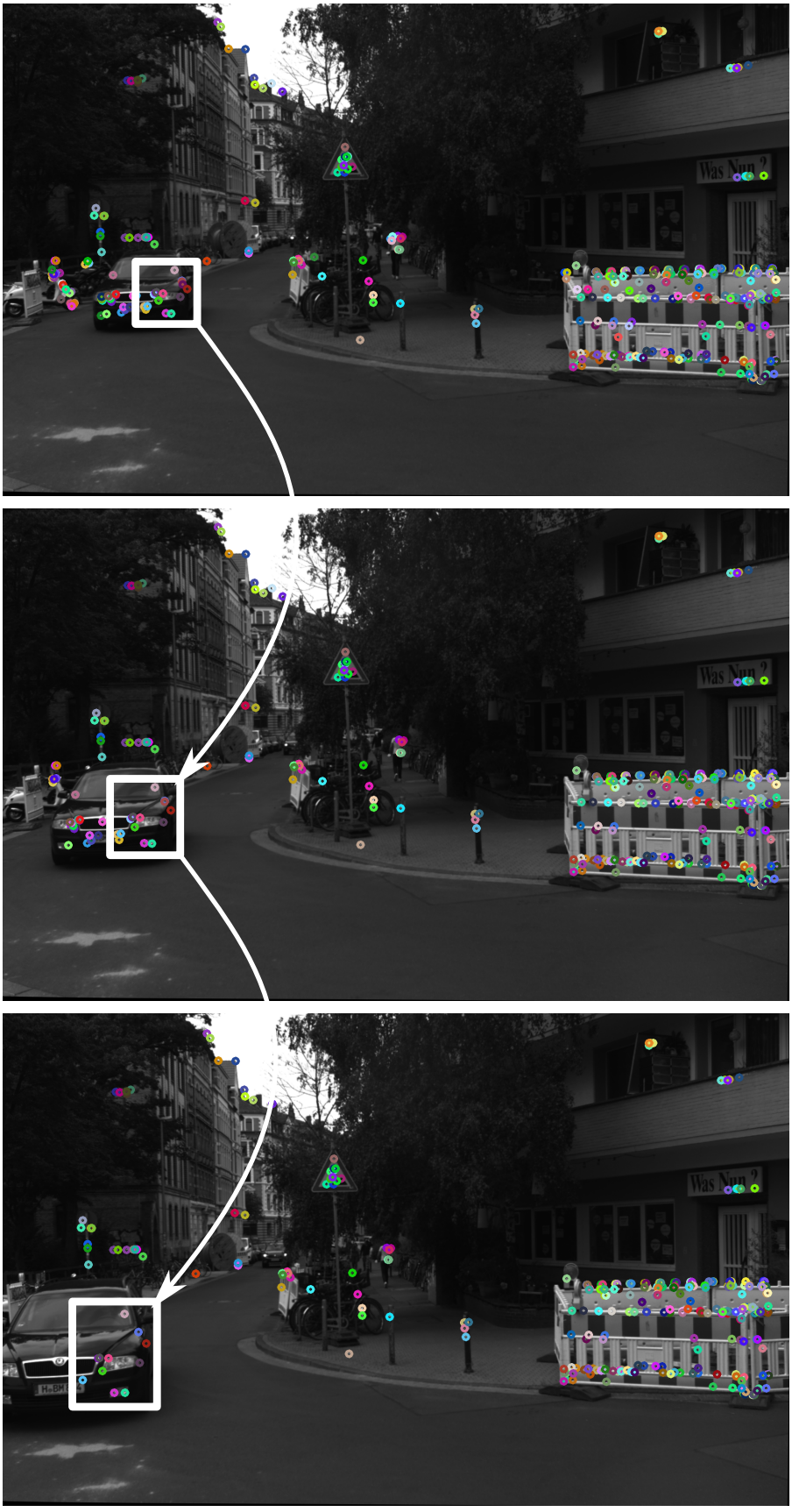


Figure 4.15: Visual feature on dynamic objects. The tracked features for an exemplary scene are shown in the left camera image. Features are consistently tracked on a dynamic vehicle approaching the sensor system. Those features are outliers and need to be removed for the ego-motion estimation.

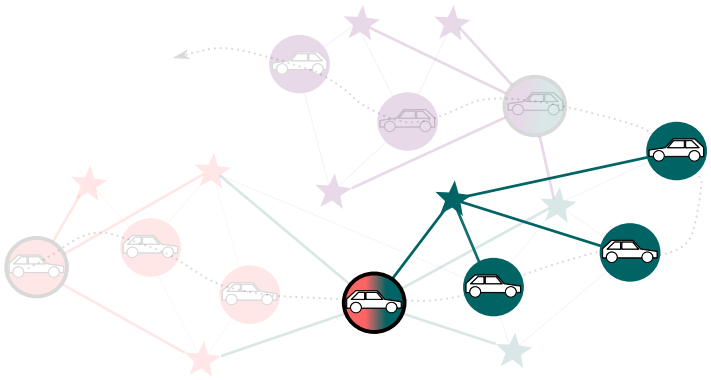


Figure 4.16: After the optimization, the residuals for each observation of each landmark in the window are checked. This is illustrated in the graph by checking the edges that are connected to the considered landmark. If at least one observation exceeds the residual threshold, the landmark is discarded from the consistent set.

at least one observation for a landmark that exceeds a maximal reprojection error. In that case, the landmark and all corresponding observations to this landmark are discarded from the consistent set of observations used for the interval-based visual odometry that is introduced in the next section.

Note that this approach heavily relies on the correct convergence of the windowed bundle adjustment. Nonetheless, due to many features and since the major part of the observations is well conditioned if the stereo camera system captures a substantial part of the static scene, this approach yields good results in practice. Furthermore, if the bundle adjustment diverges, the interval odometry will provide inconsistent results, leading to empty sets. On the bright side, if empty sets occur, we can deduce that the bundle adjustment did not converge and could not provide the needed consistent set. As a result, the user can be warned that inconsistencies were detected in the perceived data.

4.3.2 Interval-based Visual Odometry

The preselection with the robust windowed bundle adjustment only admits landmarks and observations consistent with the vehicle's estimated movement. That means we only consider landmarks with a reprojection error in the stereo images below a defined threshold with subpixel accuracy. The goal, for now, is to use the preselected landmarks to determine the relative motion taking interval observation uncertainty into account. The relative motion that we seek to determine between time step t and $t+1$ is described by ${}^{C_{l,t+1}}\mathbf{T}_{C_{l,t}}$ which consists of a rotation ${}^{C_{l,t+1}}\mathbf{R}_{C_{l,t}}$ and translation ${}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}$. Note that we utilize the RPY-Euler angle parametrization for the rotation ${}^{C_{l,t+1}}\mathbf{R}_{C_{l,t}}({}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}})$, and that we choose as a reference frame to describe the motion the left camera frame C_l. Figure 4.17 visualizes the constellation and variables we will use in the following.

First, we must check which landmarks are commonly seen at time t and $t+1$. Each landmark node connected to both pose nodes at t and $t+1$ provides constraints on the relative motion we will consider in the following. Let $\mathcal{L}$ be the set of all landmarks commonly seen at both time steps. Let the position of the i-th commonly observed landmark described in the left

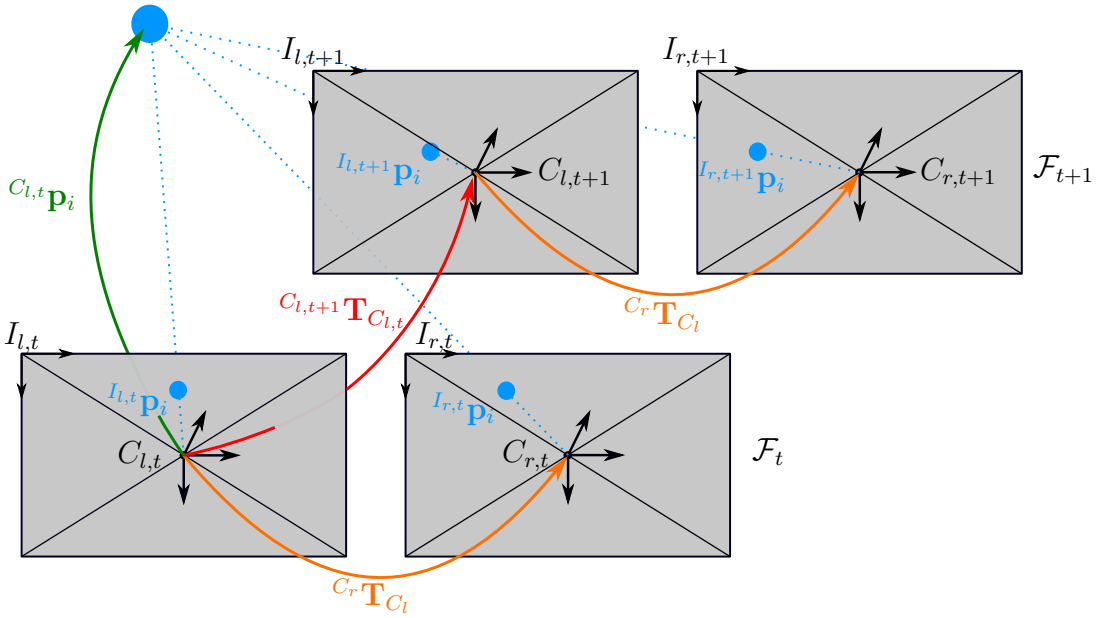


Figure 4.17: Visualization of all coordinate frames and variables used in stereo visual odometry contractor. From two stereo frames $\mathcal{F}_t$ and $\mathcal{F}_{t+1}$, the same landmark is seen. The landmark in 3D space is visualized in blue. The location of the landmark is described in the left camera frame at time t denoted by $C_{l,t}$. The coordinate frame of the right camera is denoted by $C_{r,t}$, and the transformation between the left and right camera frames is precisely known by ${}^{C_l}\mathbf{T}_{C_r}$. The frame $\mathcal{F}_{t+1}$ also consists of one stereo image pair. However, the relative motion between t and $t+1$ is defined by the transformation between the left camera frames denoted by ${}^{C_{l,t+1}}\mathbf{T}_{C_{l,t}}$ in red. The image feature locations are also colored blue. The pixel positions are described in the corresponding image coordinate frames.

camera frame at time t be ${}^{C_{l,t}}\mathbf{p}_i$. The location of the landmarks is already determined in the front-end as described in Subsection 4.2.2. Since the measured pixel position ${}^{I_{l,t}}\mathbf{p}_i$ in the left image is known from detection and tracking, we can draw the first projection constraint in the left camera image

$$ {}^{C_{l,t}}p_{z,i} \cdot \begin{pmatrix} {}^{I_{l,t}}\mathbf{p}_i \\ 1 \end{pmatrix} = \mathbf{K} \cdot {}^{C_{l,t}}\mathbf{p}_i. \tag{4.9} $$

In Figure 4.17, this constraint is visualized by the blue observation ray that goes through the projections center of the left camera in frame $\mathcal{F}_t$.

Similarly, we can draw the projection constraint for the right camera at time t as we only consider landmarks that are observed as features in the left and right images. Since the landmark position is described in the left camera frame, we first need to transform ${}^{C_{l,t}}\mathbf{p}_i$ to the right camera frame $C_{r,t}$ to apply the camera projection matrix. Since we assume a calibrated stereo camera setup, the transformation ${}^{C_r}\mathbf{T}_{C_l}$ is precisely known from calibration. As the images are also stereo rectified, we can further assume that the landmark's distance from the left and right camera is identical due to which ${}^{C_{l,t}}p_{z,i} = {}^{C_{r,t}}p_{z,i}$ holds. As a consequence, the second constraint is

$$ {}^{C_{l,t}}p_{z,i} \cdot \begin{pmatrix} {}^{I_{r,t}}\mathbf{p}_i \\ 1 \end{pmatrix} = \mathbf{K} \cdot ({}^{C_r}\mathbf{R}_{C_l} \cdot {}^{C_{l,t}}\mathbf{p}_i + {}^{C_r}\mathbf{t}_{C_l}), \tag{4.10} $$

that describes the projection of the considered landmark to the right image for time step t (see Figure 4.17).

Note that up to now, in none of the two constraints, we encounter the relative motion variables since we only focused on one time step t. However, as we also see the same landmark at $t+1$, we can consider the relative motion between the time steps t and $t+1$. Therefore, consider the left camera frame $C_{l,t+1}$ at time $t+1$. We see the same landmark in the image at the pixel position ${}^{I_{l,t+1}}\mathbf{p}_i$. To perform the projection, we first need to transform the landmark location ${}^{C_{l,t}}\mathbf{p}_i$ described in the left camera frame at time t to $C_{l,t+1}$. Therefore we need to apply the relative transformation so that we obtain

$$ {}^{C_{l,t+1}}\mathbf{p}_i = {}^{C_{l,t+1}}\mathbf{R}_{C_{l,t}} \cdot {}^{C_{l,t}}\mathbf{p}_i + {}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}. \tag{4.11} $$

Hence, the transformed point can be projected to the left image, and we should obtain the measured pixel point ${}^{I_{l,t+1}}\mathbf{p}_i$. So, we retrieve the constraint

$$ {}^{C_{l,t+1}}p_{z,i} \cdot \begin{pmatrix} {}^{I_{l,t+1}}\mathbf{p}_i \\ 1 \end{pmatrix} = \mathbf{K} \cdot ({}^{C_{l,t+1}}\mathbf{R}_{C_{l,t}} \cdot {}^{C_{l,t}}\mathbf{p}_i + {}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}). \tag{4.12} $$

In Figure 4.17, the constraint is visualized by the ray that intersects with the image plane at ${}^{I_{l,t+1}}\mathbf{p}_i$ in the left camera of frame $\mathcal{F}_{t+1}$.

As we see the same landmark also in the right image at time $t+1$, by applying again the stereo calibration parameters we obtain the projection constraint

$$ {}^{C_{l,t+1}}p_{z,i} \cdot \begin{pmatrix} {}^{I_{r,t+1}}\mathbf{p}_i \\ 1 \end{pmatrix} = \mathbf{K} \cdot \left({}^{C_r}\mathbf{R}_{C_l} \cdot ({}^{C_{l,t+1}}\mathbf{R}_{C_{l,t}} \cdot {}^{C_{l,t}}\mathbf{p}_i + {}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}) + {}^{C_r}\mathbf{t}_{C_l}\right) \tag{4.13} $$

All Stereo-Visual-Odometry (SVO) constraints can be rewritten to a multidimensional function $\mathbf{f}_i^{\mathsf{SVO}}({}^{I_{l,t}}\mathbf{p}_i, {}^{I_{r,t}}\mathbf{p}_i, {}^{I_{l,t+1}}\mathbf{p}_i, {}^{I_{r,t+1}}\mathbf{p}_i, {}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}, {}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}) = \mathbf{0}$ for one commonly observed landmark i. The set of constraints is valid for all landmarks $i \in \mathcal{L}$. As a consequence, we can formulate the CSP

$$ \mathcal{H}_{\mathsf{SVO}} : \begin{cases} \textbf{Variables: } {}^{I_{l,t}}\mathbf{p}_i, {}^{I_{r,t}}\mathbf{p}_i, {}^{I_{l,t+1}}\mathbf{p}_i, {}^{I_{r,t+1}}\mathbf{p}_i, {}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}, {}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}} \\ \textbf{Constraints: } \forall i \in \mathcal{L} : \ \mathbf{f}_i^{\mathsf{SVO}}({}^{I_{l,t}}\mathbf{p}_i, {}^{I_{r,t}}\mathbf{p}_i, {}^{I_{l,t+1}}\mathbf{p}_i, {}^{I_{r,t+1}}\mathbf{p}_i, {}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}, {}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}) = \mathbf{0} \\ \textbf{Domains: } [{}^{I_{l,t}}\mathbf{p}_i], [{}^{I_{r,t}}\mathbf{p}_i], [{}^{I_{l,t+1}}\mathbf{p}_i], [{}^{I_{r,t+1}}\mathbf{p}_i], [{}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}], [{}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}] \end{cases} . \tag{4.14} $$

Note that all rotation matrices are internally represented by Euler angles. Subsequently, we build a forward-backward contractor $\mathcal{C}_i^{\mathsf{SVO}}([{}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}], [{}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}])$ for each landmark i of the CSP $\mathcal{H}_{\mathsf{SVO}}$ and intersect all of them to obtain the final contractor

$$ \mathcal{C}_{\mathsf{SVO}}([{}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}], [{}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}]) = \bigcap_{i \in \mathcal{L}} \mathcal{C}_i^{\mathsf{SVO}}([{}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}], [{}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}]). \tag{4.15} $$

The question remains how we initialize the interval domains of the variables in the contractor (4.15). Let us first consider the landmark location. Although the landmark location ${}^{C_{l,t}}\mathbf{p}_i$ is handled as an internal set of variables, we incorporate the stereo-reconstruction results from the front-end presented in Subsection 4.2.2 directly as initial values for $[{}^{C_{l,t}}\mathbf{p}_i]$.

For setting the pixel uncertainties $[{}^{I_{l,t}}\mathbf{p}_i]$, $[{}^{I_{r,t}}\mathbf{p}_i]$, $[{}^{I_{l,t+1}}\mathbf{p}_i]$ and $[{}^{I_{r,t+1}}\mathbf{p}_i]$ we have two possibilities. One possibility is to select the maximally permitted reprojection error we defined for the preselection after the windowed bundle adjustment as the uncertainty radius. As $\mathcal{C}_i^{\mathsf{SVO}}$ is not optimal, this yields overly pessimistic results for visual odometry. The main problem of this uncertainty assessment is that we model all observations with the same highest possible error. However, some observations may provide more accurate measurements. Hence, we propose a second option for the uncertainty selection: The windowed bundle adjustment provides the remaining residuals for each observation as the reprojection error. By choosing the reprojection error as the pixel error radius for each consistent preselected landmark observation, we can assess the uncertainty of each observation individually. Hence, if an observation has a large reprojection error, the observation is handled with higher uncertainty in the interval-based odometry estimation. Furthermore, we can ensure that the residuals are consistent with at least one solution – the least squares solution of the windowed bundle adjustment. Consequently, we can interpret the interval result as the uncertainty assessment of the least squares solution under consideration of the measurement perturbation using interval tools. Note that this uncertainty selection heavily depends on the correct convergence of the bundle adjustment. Hence, we cannot guarantee that the interval-based odometry estimate will contain the true solution since the uncertainty estimation of the observations is not guaranteed to contain the error-free observation. However, this approach yields less pessimistic results in practice and provides a good estimate of the relative transformation.

The rotation domain $[{}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}]$ and translation domain $[{}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}]$ also need to be initialized. The problem that we encounter when we initialize all the domains with very large intervals is that $\mathcal{C}_{\mathsf{SVO}}$ does not contract the relative motion domains efficiently due to pessimism that mainly arises from the Euler rotation parameters. However, while the stereo-visual odometry computes a 6DOF pose, the following localization algorithms only use the 2D pose. Consequently, neither the pitch, roll, nor elevation estimation is used in the following algorithms. That is why we approximate the relative motion's roll, pitch, and elevation results by the windowed bundle adjustment results and initialize the corresponding domains with those estimates without uncertainty inflation. As a consequence, only the yaw angle $[{}^{C_{l,t+1}}\varphi_{C_{l,t}}]$ and the position $(\,[{}^{C_{l,t+1}}t_{x,C_{l,t}}]\;\;[{}^{C_{l,t+1}}t_{y,C_{l,t}}]\,)^\intercal$ are initialized with large intervals. We store the odometry data determined by $\mathcal{C}_{\mathsf{SVO}}$ to each frame $\mathcal{F}_{t+1}$ at $t+1$ as the transformation from $\mathcal{F}_t$ to $\mathcal{F}_{t+1}$.

4.3.3 Keep the SLAM-Graph Clean

A clean SLAM-graph containing as little as possible and as much as necessary information is vital for efficiency. Landmark observations that do not contribute valuable information for relative motion computations are removed from the graph. Specifically, landmarks that are seen only once, that are connected to only one pose node, are deleted. This often happens with keyframes, as new features/landmarks are detected only for keyframes, but not all of them are accurately tracked in subsequent frames. To maintain a clean graph, a separate process identifies and removes such landmarks. By deleting these irrelevant landmarks, the graph analysis becomes faster and memory consumption is reduced.

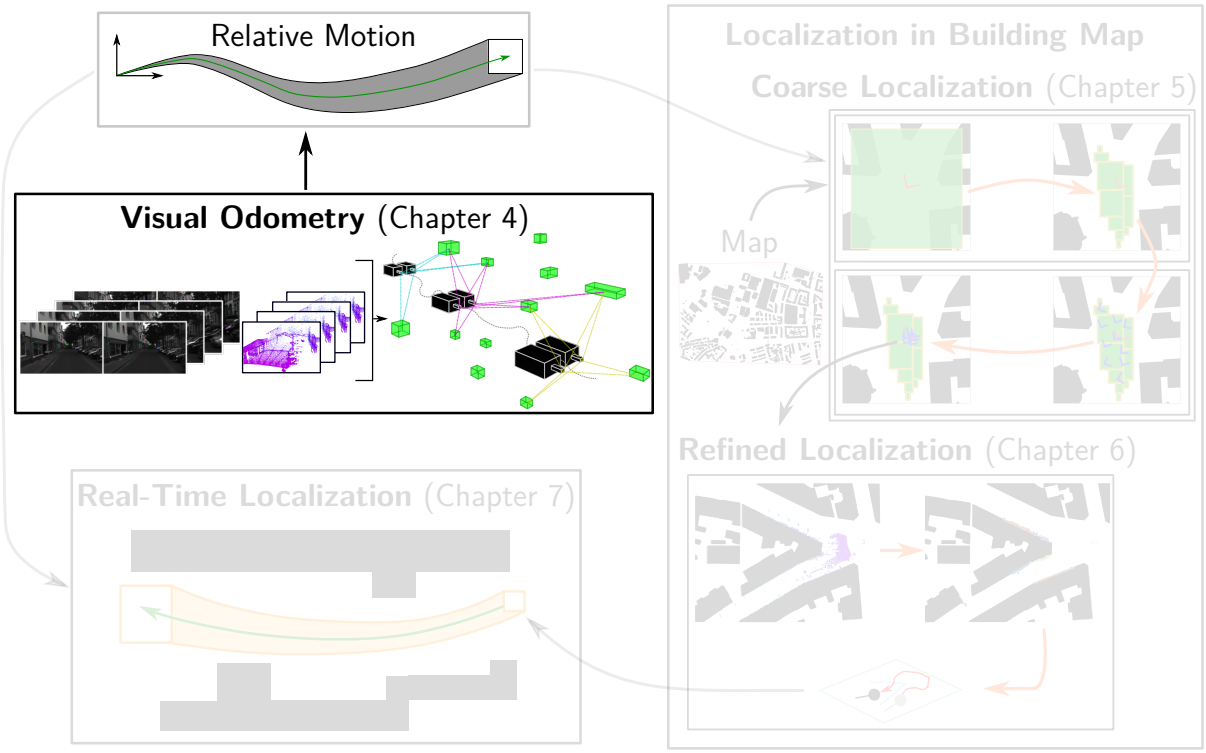


Figure 4.18: The visual odometry module provides the relative motion defined in a local reference frame. We must determine the spatial link between the local reference and the map frame. The next chapter solves this problem.

4.4 Module Output

The visual odometry module computes the relative motion of the vehicle and stores for each frame $\mathcal{F}_{t+1}$ the relative transformation ${}^{C_{l,t+1}}\mathbf{T}_{C_{l,t}}$. The relative transformation is internally represented by a translation vector ${}^{C_{l,t+1}}\mathbf{t}_{C_{l,t}}$ and euler angles ${}^{C_{l,t+1}}\boldsymbol{\xi}_{C_{l,t}}$. While the roll ${}^{C_{l,t+1}}\psi_{C_{l,t}}$, pitch ${}^{C_{l,t+1}}\theta_{C_{l,t}}$ and elevation ${}^{C_{l,t+1}}t_{z,C_{l,t}}$ are handled as correctly estimated values from the bundle adjustment, the yaw orientation $[{}^{C_{l,t+1}}\varphi_{C_{l,t}}]$ and the 2D position $([{}^{C_{l,t+1}}t_{x,C_{l,t}}]\ [{}^{C_{l,t+1}}t_{y,C_{l,t}}])^\intercal$ is represented by intervals taking the observation uncertainty of the locally seen landmarks as ORB-features into account.

Using the hybrid visual odometry approach as a stand-alone algorithm has two problems. First, since we store the relative motion in a dead-reckoning fashion for each frame $\mathcal{F}_t$, the reconstruction of the relative motion to a reference frame leads to uncertainty accumulation. We illustrate this in Figure 4.18 using the trajectory tube with rising uncertainty. Global information is necessary to tackle this problem to reduce accumulating uncertainty. In this work, we will use building maps. However, the second problem is that visual odometry only provides the relative motion with respect to a predefined local reference frame. That means we cannot know a priori the relative location and orientation of the odometry frame concerning the map frame. This problem is also known as initial localization or global localization problem. The next chapter introduces our interval-probabilistic coarse localization that solves this problem.

5 Coarse Localization

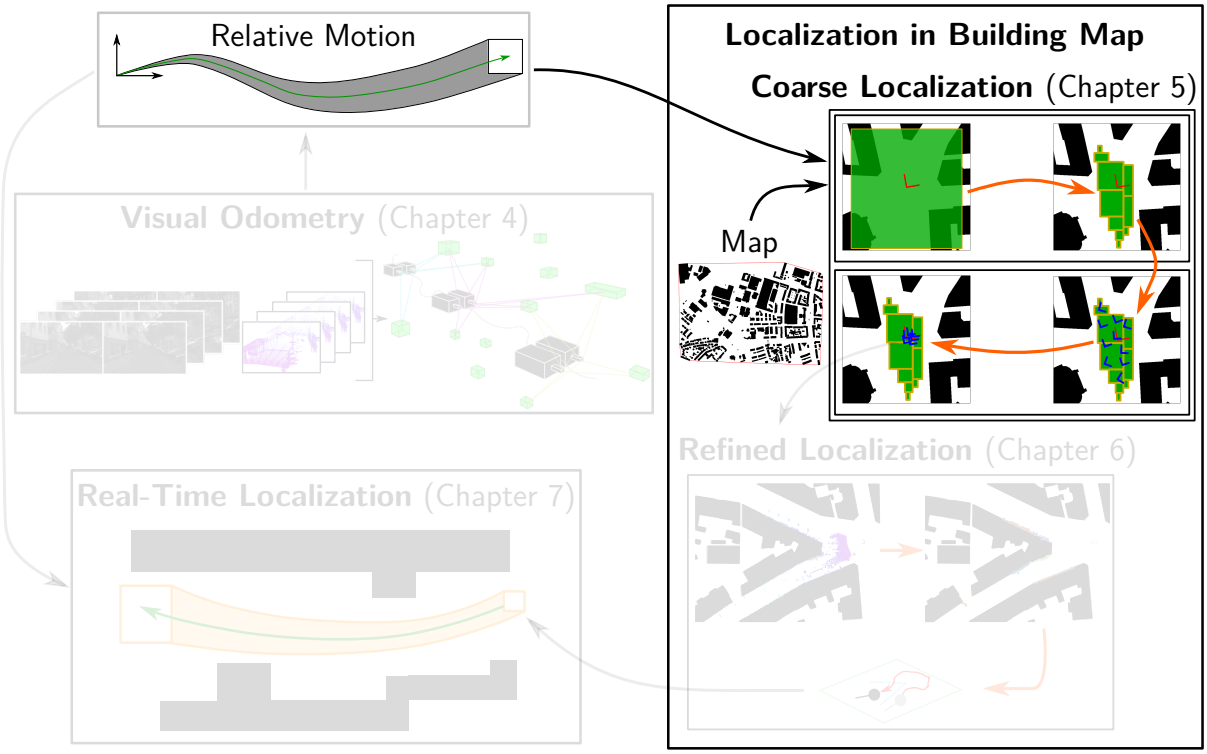


Figure 5.1: The coarse localization is the second module in the localization pipeline and uses the ego-motion estimation determined by the visual odometry.

In the previous chapter, the visual odometry module was introduced that provides the ego-motion estimation of the vehicle in a dead-reckoning fashion. However, our goal is to localize the vehicle on the map. Since the ego-motion of the vehicle is described in a local reference frame, we do not know the spatial correspondence between the local reference and the map frame in which the buildings are described. This chapter presents a hybrid interval-probabilistic localization under large global positioning uncertainties in urban environments.

Many localization solutions in the literature address the tracking problem. However, the problem we seek to solve is fundamentally different: If the initial pose uncertainty is high, the local measurements' association with elements in the map may become inapplicable since matches can be ambiguous. Tracking each possible match will let the runtime explode. Therefore, we need to approach the problem differently. In the literature, localization under

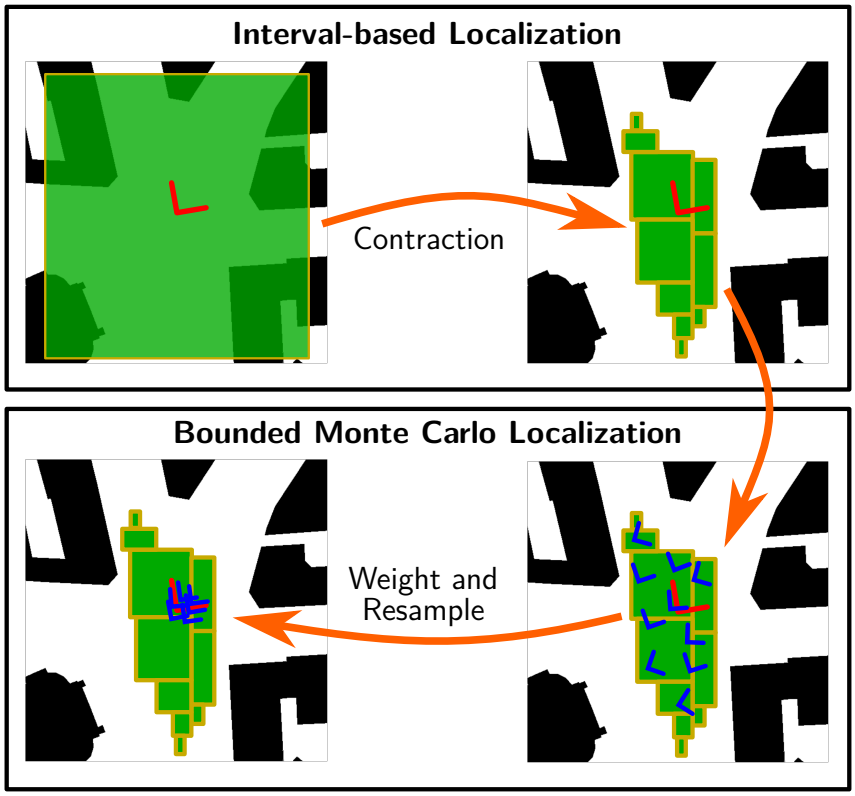


Figure 5.2: Coarse hybrid interval-probabilistic localization. Buildings in the maps are visualized in black. The correct pose is colored red. The coarse interval-based localization provides the feasible set of poses (green). The bounded MCL spreads particles inside the feasible set and determines the most likely solution by an aggressive resampling procedure.

large uncertainties is typically called global localization. However, the term *global* implies that no information on the vehicle's initial location is known. Hence, the typical global localization problem considers a map where we must assume that the vehicle can be everywhere. In contrast, in our case, we do have information on the vehicle position from GNSS measurements. Nonetheless, global positioning systems may provide very inaccurate pose estimates due to occlusions and/or multi-path effects, especially in urban canyons. In the worst case, no global information on the robot's pose is available. The localization problem we are addressing in this chapter can be categorized between global localization and local tracking. We are not performing global localization since we have a rough initial estimate of the vehicle's location from GNSS data. However, the uncertainty of the GNSS data is so large that an unambiguous matching of locally seen landmarks is impossible, due to which multiple hypotheses need to be tracked. The dominant approach in the literature to solve that problem are derivatives of the Monte Carlo Localization (MCL). In this work, we introduce a real-time capable set-membership enhanced bounded MCL scheme that compensates for the shortcomings of classical MCL approaches, improving the convergence and reducing the runtime. Our method efficiently solves the localization problem using a LiDAR sensor, odometry data, GNSS data, and publicly available building maps provided by OSM and LOD2 city model.

As presented in the overview in Figure 5.2, our method consists of two steps. Using interval analysis, we first narrow down the feasible set of poses under consideration of the local

LiDAR data, the vehicle odometry provided by the visual odometry module, and occasionally available uncertain GNSS measurements. This is illustrated in the upper block in Figure 5.2 and represents the interval-based localization method. The result of the interval-based method – the feasible set of poses – is used for the second method illustrated in the bottom block. We perform a bounded MCL within the feasible set to determine the most likely solution. By tracking the feasible solution set with the interval method, only a few particles are necessary to determine the most likely candidates in the feasible set. An aggressive resampling of the particles is possible, leading to faster convergence to the correct solution. Our hybrid interval-probabilistic approach was already published in [141] and will be presented here in more detail.

After introducing the task description, notations, and assumptions, we present in Section 5.2 the first part of our hybrid method – the interval-based localization – that determines the feasible set of poses. Then, in Section 5.3, we introduce the bounded MCL that determines the most likely solutions within the feasible set.

5.1 Task Description, Notations and Assumptions

The robot is initially placed somewhere on the building map, but it lacks knowledge of its whereabouts [18, p. 194]. From GNSS readings, we have a rough estimate of its position. However, this position estimate can have large uncertainty. The goal is to determine the set of all feasible poses ${}^{M}\hat{\mathbf{T}}_{L_t} \in {}^{M}\mathcal{P}_{L_t}$ of the vehicle in the given map with building footprints and the most likely poses within the feasible set. We choose the vehicle fixed LiDAR coordinate frame L_t at time t as the reference frame for the localization of the vehicle. We consider three basic assumptions: First, the robot can never be inside a building. Second, the robot will not exit the map. Third, GNSS position estimates have a maximum bounded error, and the vehicle has to be located within the uncertainty region of the GNSS measurements. As a result, we exclude buildings from our map that consist of bridge-like structures. As measurements, we use odometry and LiDAR data locally, while we also occasionally have very uncertain GNSS measurements on the position.

We perform the localization in a map that consists of building footprints in 2D. The robot pose ${}^{M}\hat{\mathbf{T}}_{L_t} = \begin{pmatrix} {}^{M}\mathbf{t}_{L_t} & {}^{M}\varphi_{L_t} \end{pmatrix}^T$ consists of two translation parameters ${}^{M}\mathbf{t}_{L_t} = \begin{pmatrix} t_x & t_y \end{pmatrix}^T$ and one orientation parameter ${}^{M}\varphi_{L_t}$. When we refer to a set of poses ${}^{M}\mathcal{P}_{L_t}$, we imply that the three parameters inflate to sets represented by intervals. Hence, the set of poses ${}^{M}\mathcal{P}_{L_t} = \begin{pmatrix} {}^{M}\mathcal{T}_{L_t} & {}^{M}\Phi_{L_t} \end{pmatrix}^T$ consists of a set of translations ${}^{M}\mathcal{T}_{L_t}$ and a set of orientations ${}^{M}\Phi_{L_t}$. A connected set of orientations is well defined by an interval ${}^{M}\Phi_{L_t} = [{}^{M}\varphi_{L_t}]$. Due to ambiguities in localization, the feasible set of orientations is not necessarily connected. That is why we approximate ${}^{M}\Phi_{L_t}$ by a subpaving ${}^{M}\Phi^{\boxplus}_{L_t}$, that is a union of non-overlapping intervals $[{}^{M}\varphi_{L_t}] \subset {}^{M}\Phi^{\boxplus}_{L_t}$. Similarly, the set of translations ${}^{M}\mathcal{T}_{L_t}$ is also approximated by an outer subpaving ${}^{M}\mathcal{T}^{\boxplus}_{L_t}$, that consists of non-overlapping translation boxes $[{}^{M}\mathbf{t}_{L_t}] \subset {}^{M}\mathcal{T}^{\boxplus}_{L_t}$ with $[{}^{M}\mathbf{t}_{L_t}] = \begin{pmatrix} [t_x] & [t_y] \end{pmatrix}^T$, where $[t_x]$ is the interval in the x and $[t_y]$ in the y direction. The rotation and translation subpaving define the pose. Hence, the set of poses ${}^{M}\mathcal{P}^{\boxplus}_{L_t}$ is represented by a 3D subpaving.

A possible translation subpaving ${}^{M}\mathcal{T}^{\boxplus}_{L_t}$ is illustrated Figure 5.3. Note that the border of the

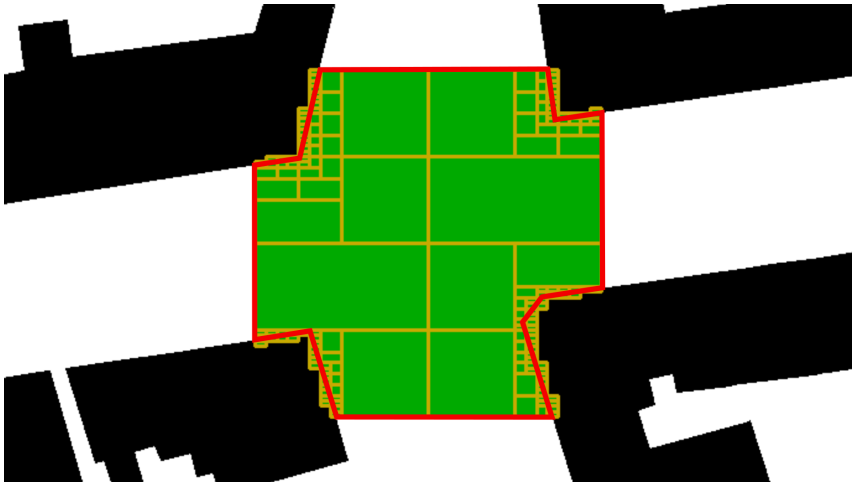


Figure 5.3: Translation subpaving. The buildings are shown in black. ${}^{M}\mathcal{T}_{L_t}^{\boxplus}$ is a set of non-overlapping green boxes $[{}^{M}\mathbf{t}_{L_t}] \subset {}^{M}\,\mathcal{T}_{L_t}^{\boxplus}$. The subpaving ${}^{M}\mathcal{T}_{L_t}^{\boxplus}$ approximates the set of feasible translations. A red polygon illustrates the border of the correct feasible set.

set we want to describe is colored red. Due to the approximation of this set with a subpaving, we overestimate this set. That is the reason why there are green parts beyond the red border. However, we can increase the accuracy of the approximation by reducing the size of the minimal permitted box width. On the downside, this will lead to more boxes. Consequently, the computational effort will also increase as each box needs to be processed. Since we use within this chapter a subpaving representation of the set of poses ${}^{M}\mathcal{P}_{L_t}^{\boxplus}$, we will omit the subpaving qualifier $\boxplus$ in the following.

5.1.1 Odometry

The hybrid interval-probabilistic visual odometry presented in Chapter 4 provides the odometry data and the corresponding uncertainty. Nonetheless, any method that provides such data is applicable. Given the pose ${}^{M}\hat{\mathbf{T}}_{L_t}$ at time t and the odometry data ${}^{L_t}\hat{\mathbf{T}}_{L_{t+1}} = \begin{pmatrix}\Delta\mathbf{t} & \Delta\varphi\end{pmatrix}^{T}$, the pose at time $t+1$ is determined by

$$\mathbf{f}_{\text{odom}}({}^{M}\hat{\mathbf{T}}_{L_t}) = {}^{M}\,\hat{\mathbf{T}}_{L_{t+1}} = \begin{pmatrix} {}^{M}\mathbf{t}_{L_t} + \begin{pmatrix}\cos{}^{M}\varphi_{L_t} & -\sin{}^{M}\varphi_{L_t} \\ \sin{}^{M}\varphi_{L_t} & \cos{}^{M}\varphi_{L_t}\end{pmatrix} \cdot \Delta\mathbf{t} \\ {}^{M}\varphi_{L_t} + \Delta\varphi \end{pmatrix}. \tag{5.1}$$

Note that we assume the odometry data ${}^{L_t}\hat{\mathbf{T}}_{L_{t+1}}$ to be uncertain. Hence, the odometry parameters are represented by intervals $\Delta\varphi \in [\Delta\varphi]$ and $\Delta\mathbf{t} \in [\Delta\mathbf{t}]$ and the mathematical operators are extended to interval arithmetic.

5.1.2 Map

The used building maps are already introduced in Section 2.4. However, in the scope of this chapter, we need to introduce further properties of the building footprint map. First, we need to define the borders of a map to determine the inner and outer regions. We define the map

by the convex hull over all buildings. Note that the borders of a map define a closed region. As a result, the border can be represented by a polygon. Furthermore, we represent a building in the map also by a polygon. A polygon P is described by an ordered set of corner points $\mathcal{C}_P = \{{}^M\mathbf{c}_1, \ldots, {}^M\mathbf{c}_n\}$ with ${}^M\mathbf{c}_i = \begin{pmatrix} x & y \end{pmatrix}^T$ described in the map coordinate frame for $n \in \mathbb{N}$, $i \in \{0, \ldots, n\}$ and $x, y \in \mathbb{R}$ so that the border of the polygon is defined by the line segments between the corner points. We also consider the map uncertainty by inflating the corner points to boxes in our computations.

5.1.3 Uncertain GNSS Position

The GNSS receiver provides the latitude and longitude position described in WGS84 as introduced in Subsection 2.3.3. As the building map is globally referenced and has a locally defined frame, we transform the GNSS position from WGS84 to the map frame. We experienced inconsistencies in the uncertainty estimate of the GNSS data and therefore decided to consider a large empirically determined maximum error bound of $50\,\mathrm{m}$ of the transformed position estimate. However, the selection of this maximal error limit depends on the used GNSS receiver. Note that the maximal error bound does not have to be static. If sophisticated error models on the GNSS reliability are available, also dynamically changing error bounds can be used. However, we will stick to a fixed upper-bound error for our work.

5.2 Interval-based Localization

Initially, we do not know the vehicle's position and orientation on the map. We can represent the orientation interval $[-\pi, \pi]$ by a subpaving. That means we divide the interval into smaller subsets. We independently narrow down the feasible set of positions for each orientation subset by applying multiple contractors.

To introduce our interval-based localization approach, we first want to focus on one orientation subset that encloses the correct orientation. Later, we will generalize our explanation to an arbitrary orientation subset. If the initial orientation is known up to a defined uncertainty, the localization problem simplifies finding the two translation parameters. According to the assumptions, the robot's position can be described by a set of positions on the map with an empty intersection with the buildings. The left figure in Figure 5.4 illustrates such a set for the initial pose: The set of all feasible positions is colored green while the buildings are colored black. Initially, the robot can be located everywhere in the green region. The green region – the feasible set of positions – is represented by a subpaving. When the robot moves, the subsets of the subpaving are updated according to the measured odometry. However, the robot can never be located inside the buildings also during movement. That means when the robot moves, we can discard those regions that are shifted into the building regions. As a result, the feasible set of positions is gradually narrowed down as more and more parts of the feasible region get discarded when the robot moves. This is illustrated in the middle and right figure of Figure 5.4. We call the contractor that contracts the feasible position region so that only those parts remain that do not overlap with the buildings, the *No-Overlap Contractor* which will be presented in Subsection 5.2.3.

Figure 5.4: Localization sequence with correct initial orientation. The buildings are black and the set of all feasible vehicle positions is colored green. The ground truth pose is visualized red. The images show the localization result of a sequence starting on the left and ending on the right. The coarse localization is initialized in the left figure.

For the No-Overlap Contractor, only the odometry data as local data is used. However, also the LiDAR data enables us to draw constraints for the feasible set of poses: Due to the height and solid structure of building facades in urban environments, almost all laser beams are blocked – that means the visibility region is restricted – by the building walls. Somehow, there are exceptions: Open doors and non-solid windows allow laser beams to capture measurements beyond the walls. However, LiDAR measurements that go through small openings lead to small structures in the LiDAR scan that can be filtered out. For the filtered LiDAR data, the constraint that measurements cannot cross walls becomes valid. Using the no-cross constraint, we can further contract the feasible set of poses. We call this contractor the *No-Cross Contractor*, which will be presented in Subsection 5.2.4.

As the No-Overlap and No-Cross Contractor solely use local sensor data, multiple disconnected sets for the feasible pose can occur. GNSS positioning information provides further information to reduce the feasible set. However, one of the major challenges of GNSS data is its inaccuracy in urban canyons. Nonetheless, we can provide pessimistic but always valid upper bounds on the error of the position estimate. Based on the upper bound of the position measurement error, we define a circular GNSS-uncertainty region where the vehicle must be located at the time of measurement. Based on this constraint, we can further contract the feasible region as presented in Subsection 5.2.5.

Up to this point, we assumed to know the orientation up to some uncertainty. Now, let us withdraw this assumption to generalize our approach. In that case, we can represent the orientation interval $[-\pi, \pi]$ by a subpaving. That means we divide the interval into smaller subsets. We can apply the same contraction algorithms described above for each of those orientation subsets. Incorrect initial orientation subsets will lead to an empty set for the translation, as all translation subsets will become infeasible. If so, those orientation subsets are dismissed from the feasible orientation set. Applying the contractors to each translation subpaving of each orientation subset may become computationally expensive. Fortunately, the

(a) Map. (b) Initialization. (c) Initial Subpaving.

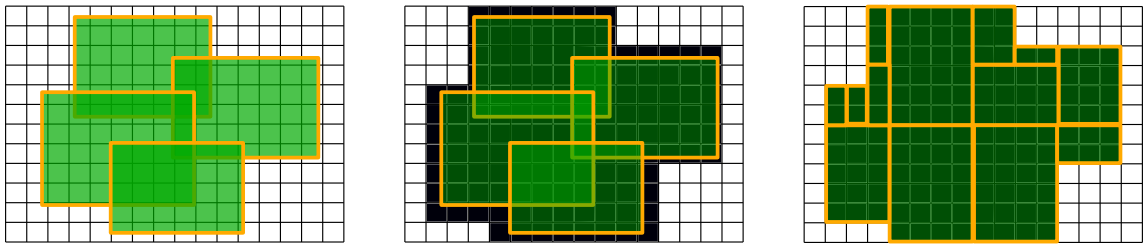
Figure 5.5: Exemplary Scenario. Figure 5.5a depicts the map and the trajectory. The robot has a straight linear movement. The initial set of poses is illustrated in Figure 5.5b. Figure 5.5c shows the subpaving after applying the No-Overlap Contractor. The correct pose is visualized in red.

translation subpavings of each orientation subset can be handled independently. Consequently, to achieve real-time performance, we apply parallel execution of the contractors by multi-threading the tasks on a consumer-grade CPU. Each thread performs the contraction of the translation subpaving of one orientation subset. If orientation subsets, for which the translation becomes empty, are dismissed, we bisect a non-empty orientation subpaving and process it with the released thread. The rotation splitting further passively contracts the orientation subpaving, making a balanced workload possible. We present the rotation splitting in Subsection 5.2.6.

To illustrate our approach, we want to introduce a simple scenario depicted in Figure 5.5a. In this simple scenario, we assume the vehicle only drives forward along a straight line, as illustrated with the red arrow. In the following, we will explain our contractors based on this example.

5.2.1 Initialization

The robot can have an arbitrary initial translation and orientation. We set the initial set of orientations Φ_0 to $[-\pi, \pi]$ and the initial translation subpaving $\mathcal{T}_0$ to the whole map as illustrated in Figure 5.5b. For the sake of simplicity, we will omit the frame indices M and L as all poses refer to the pose of the LiDAR frame L in the map frame M. We will only keep the time index. We subdivide Φ_0 into q equally sized subsets $\Phi_{i,0}$ with $i \in \{1, ..., q\}$. That means each subset $\Phi_{i,0}$ generates an independent set of poses $\mathcal{P}_{i,0} = \begin{pmatrix} \mathcal{T}_0 & \Phi_{i,0} \end{pmatrix}^T$. As a consequence, we obtain q independent potential initial pose sets, which have the same initial translation subpaving $\mathcal{T}_0$ but differ concerning the set of initial orientations $\Phi_{i,0}$. The following algorithms for contracting and updating the set of poses are applied to each initial pose set $\mathcal{P}_{i,0}$ independently. The subdivision of rotation parameters can also be interpreted as generating multiple rotation bins that must be considered independently. The computations are performed in parallel on multiple threads. Directly after the initialization, the contractors are applied to dismiss infeasible poses.



(a) Non-regular subpaving. (b) Raster coloring. (c) Regularized subpaving.

Figure 5.6: Regularization of a 2D subpaving with the image contractor. Figure 5.6a shows a non-regular subpaving. First, as illustrated in Figure 5.6b, a raster is defined in which cells with a non-empty intersection with the subpaving are colored. A regular subpaving, as shown in Figure 5.6c, is determined by applying the image contractor to the colored raster.

5.2.2 Pose Update with Odometry Data

We use the odometry data to update the set of feasible poses for each time step t as presented in Subsection 5.1.1. Let $\mathcal{P}_{i,t} = \begin{pmatrix}\mathcal{T}_{i,t} & \Phi_{i,t}\end{pmatrix}^T$ and $\mathcal{P}_{i,t+1} = \begin{pmatrix}\mathcal{T}_{i,t+1} & \Phi_{i,t+1}\end{pmatrix}^T$ be the set of feasible poses at time t and $t+1$ described in the map frame. By applying the odometry update described in (5.1) to each subset of the subpaving $\mathcal{T}_{i,t}$, the updated set of positions $\mathcal{T}_{i,t+1}$ is obtained. Therefore, the operations in (5.1) are extended to interval arithmetic so that ${}^M\mathbf{t}_{L_t} \in \mathcal{T}_{i,t}$, ${}^M\mathbf{t}_{L_{t+1}} \in \mathcal{T}_{i,t+1}$, ${}^M\varphi_{L_t} \in \Phi_{i,t}$, ${}^M\varphi_{L_{t+1}} \in \Phi_{i,t+1}$, holds.
However, the direct update step causes a regular position subpaving $\mathcal{T}_{i,t}$ to be updated to a non-regular subpaving $\mathcal{T}_{i,t+1}$. This is because the odometry uncertainty inflates the pose estimate, and the subsets may start overlapping. An exemplary non-regular subpaving is illustrated in Figure 5.6a. There are two possibilities to obtain the regular subpaving to the feasible set of poses. One possibility is to reformulate the pose update as a set-inversion problem. Alternatively, the image contractor can be employed to regularize the subpaving after the direct odometry update. In the following, we will present both possibilities.

Odometry Update by Set-Inversion

The set-inversion approach does not directly apply the odometry update as presented in (5.1). Instead, the inverse of $\mathbf{f}_{\text{odom}}(\hat{\mathbf{T}}_{i,t})$ is employed to determine a regular subpaving for $\mathcal{P}_{i,t+1}$ from $\mathcal{P}_{i,t}$. The recursive algorithm is presented in Algorithm 5.

Initially, we set all domains of $\mathcal{P}_{i,t+1}$ to large intervals. We apply the inverse odometry function

$$\mathbf{f}^{-1}_{\text{odom}}(\hat{\mathbf{T}}_{t+1}) = \hat{\mathbf{T}}_t = \begin{pmatrix} \mathbf{t}_{t+1} - \begin{pmatrix}\cos(\varphi_{t+1} - \Delta\varphi) & -\sin(\varphi_{t+1} - \Delta\varphi) \\ \sin(\varphi_{t+1} - \Delta\varphi) & \cos(\varphi_{t+1} - \Delta\varphi)\end{pmatrix} \cdot \Delta\mathbf{t}_t \\ \varphi_{t+1} - \Delta\varphi \end{pmatrix}, \tag{5.2}$$

to obtain $\hat{\mathcal{P}}_{i,t}$ according to line 2. As we already know $\mathcal{P}_{i,t}$ represented by a subpaving from the previous iteration step, we contract $\hat{\mathcal{P}}_{i,t}$ by considering the intersection with $\mathcal{P}_{i,t}$. After the contraction of $\hat{\mathcal{P}}_{i,t}$, we apply the forward odometry function $\mathbf{f}_{\text{odom}}({}^M\hat{\mathbf{T}}_{L_t})$ to determine $\mathcal{P}_{i,t+1}$

Algorithm 5: Odometry Update by Set-Inversion.

Data: $\mathcal{P}_{i,t} = \begin{pmatrix} \mathcal{T}_{i,t} & \Phi_{i,t} \end{pmatrix}^T$ and $\mathcal{P}_{i,t+1} = \begin{pmatrix} [-\infty, +\infty] & [-\infty, +\infty] & [-\pi, \pi] \end{pmatrix}^T$
Result: $\mathcal{P}_{i,k+1}$
Function *OdometryUpdateBySetInversion($\mathcal{P}_{i,t}$,$\mathcal{P}_{i,t+1}$)*
  $\hat{\mathcal{P}}_{i,t} = [\mathbf{f}_{\mathsf{odom}}^{-1}](\mathcal{P}_{i,t+1})$;
  $\hat{\mathcal{P}}_{i,t} = \hat{\mathcal{P}}_{i,t} \cap \mathcal{P}_{i,t}$;
  $\mathcal{P}_{i,t+1} = \mathcal{P}_{i,t+1} \cap [\mathbf{f}_{\mathsf{odom}}](\hat{\mathcal{P}}_{i,t})$;
  $w = \mathsf{getSmallestTranslationWidth}(\mathcal{P}_{i,t+1})$;
  if $w > \epsilon$ **then**
    $(\mathcal{P}_{\mathsf{left}}, \mathcal{P}_{\mathsf{right}}) = \mathsf{bisect}(\mathcal{P}_{i,t+1})$;
    $\mathcal{P}_{i,t+1} = \mathcal{P}_{i,t+1} \cup \mathsf{OdometryUpdateBySetInversion}(\mathcal{P}_{i,t}, \mathcal{P}_{\mathsf{left}})$;
    $\mathcal{P}_{i,t+1} = \mathcal{P}_{i,t+1} \cup \mathsf{OdometryUpdateBySetInversion}(\mathcal{P}_{i,t}, \mathcal{P}_{\mathsf{right}})$;
  end
  return $\mathcal{P}_{i,t+1}$;

from $\hat{\mathcal{P}}_{i,t}$. Next, we bisect the translational parameter with the highest interval width of $\mathcal{P}_{i,t+1}$. So we obtain a left and right child $\mathcal{P}_{\mathsf{left}}$ and $\mathcal{P}_{\mathsf{right}}$. The same procedure is recursively applied to the obtained left and right children until the translation parameters of $\mathcal{P}_{i,t+1}$ are smaller than a given threshold ϵ. The set of all obtained pose estimates $\mathcal{P}_{i,t+1}$ for all $i \in \{1, \ldots, q\}$ after applying the recursive algorithm provides the regular subpaving $\mathcal{P}_{t+1}$.

Direct Odometry Update and Regularization with the Image Contractor

How we employ the image contractor to regularize a subpaving is illustrated in Figure 5.6. After applying a sequence of uncertain odometry updates to the set of feasible poses, the translation subpaving will consist of subsets that overlap as illustrated in Figure 5.6a. First, we insert the subpaving into a raster which we interpret as an image. The scale of a pixel can be chosen arbitrarily. In our experiments, we choose a scale of $0.5 \, \frac{\mathsf{m}}{\mathsf{pix}}$. As shown in Figure 5.6b, we approximate the subpaving by coloring those pixels with a non-empty intersection with the subpaving accordingly. As a result, we obtain a binary image. Using that binary image, we apply the image contractor in a SIVIA fashion. First, we start with an arbitrarily large box that includes all colored pixels and apply the image contractor to that box. The box is contracted to the box hull containing all colored pixels. Due to the arbitrary shape of the original subpaving, the box hull will also contain non-colored pixels. Hence, we bisect the box and apply the contractor again on those newly generated boxes. We recursively apply this algorithm until none of the generated boxes contain non-colored pixels. This can be detected very efficiently using the integral image. If we apply this SIVIA algorithm to the binary image shown in Figure 5.6b, we will retrieve the subpaving shown in Figure 5.6c. Hence, the regularization can be computed very efficiently.
On the downside, the new subpaving overestimates the original set due to the discretization with a raster. Due to the overestimation problem, we only apply the image contractor if the overlap area of the subsets exceeds a given threshold for the translation subpaving.

The experimental evaluation shows that the regularization with the image contractor performs better than the set-inversion approach. This is because the second option makes extensive

use of parallel computation possible. While the set-inversion approach needs to perform the computations sequentially, the direct pose update can be performed in parallel. Building the raster and applying the image contractor introduces some computational overhead. However, compared to the sequential computation of the set-inversion approach, the image contractor-based regularization is still computationally more efficient. That is why we use the second option to regularize the subpaving of the translational parts of the feasible set of poses with the image contractor.

5.2.3 No-Overlap Contractor – Vehicle Outside Buildings

Given a set of poses $\mathcal{P}_i = \begin{pmatrix} \mathcal{T}_i & \Phi_i \end{pmatrix}^T$, the goal is to delete those parts of the translation subpaving that overlap with buildings. In other words, we restrict the feasible set only to the free space not occupied by buildings. Therefore, we apply the *No-Overlap Contractor*. The algorithm is depicted in Algorithm 6. Note that $\mathcal{P}_i =^M \mathcal{P}_{L_t,i}$ holds and that we omit the frame indices and the time index, as we only consider the LiDAR frame pose in the map frame for one time step t.

Let $[\mathbf{t}] \subset \mathcal{T}_i$ be a subset of the subpaving. First, we determine all buildings $\mathcal{B}^*$ that are in the vicinity of $[\mathbf{t}]$ as shown in Algorithm 6 in line 6. For each building B we apply the polygon separator $\mathcal{S}_{\text{poly}}([\mathbf{t}], B)$ as introduced in Part 2.2.2.2 on $[\mathbf{t}]$. As a result, the separator provides an inner box $[\mathbf{t}_{\text{in}}]$ and an outer box $[\mathbf{t}_{\text{out}}]$. Since $[\mathbf{t}_{\text{out}}]$ defines the box hull over all parts of $[\mathbf{t}]$ that are outside B, the wrapping leads to an overestimation. Hence, parts inside B can also be part of the hull. That is why we first determine the intersection $[\mathbf{t}_{\text{inter}}] = [\mathbf{t}_{\text{out}}] \cap [\mathbf{t}_{\text{in}}]$. If the intersection is empty, then we can indeed replace $[\mathbf{t}]$ by $[\mathbf{t}_{\text{out}}]$. If the intersection is non-empty, $[\mathbf{t}_{\text{out}}]$ can contain inner parts of B. If the smallest width of $[\mathbf{t}_{\text{out}}]$ is larger than a given threshold ϵ, we will bisect $[\mathbf{t}_{\text{out}}]$ and apply the separator recursively (SIVIA). As a result, for $[\mathbf{t}]$, we will obtain multiple subsets that represent the outer part. Accordingly, we replace the subset $[\mathbf{t}]$ in the subpaving $\mathcal{T}_i$ by the obtained outer subsets. In Algorithm 6, the stack data structure may also be replaced by a queue.
We can proceed similarly regarding the map hull. The only difference is that only subsets inside the hull should be considered. As a result, we can use a similar algorithm as presented in Algorithm 6. The main difference is that instead of inserting and bisecting $[\mathbf{t}_{\text{out}}]$, we need to apply the steps to $[\mathbf{t}_{\text{in}}]$.
When we apply the No-Overlap Contractor on the initial set of poses as depicted in Figure 5.5b, the output subpaving that only contains the feasible subsets is shown green in Figure 5.5c. In Figure 5.7, we can see the result after applying the sequence of pose updates with odometry data and no-overlap contractions for different orientations. Please note the white "shadows" of the buildings in the green feasible sets $\mathcal{P}_a$, $\mathcal{P}_b$ and $\mathcal{P}_c$ in the figures 5.7a, 5.7b and 5.7c depending on the different orientation Φ_a, Φ_b and Φ_c.

5.2.4 No-Cross Contractor – LiDAR Data in Visible Region

The LiDAR provides information on the visibility of the scene. If a specific part of the scene reflects a laser beam, we can deduce the information that between the measured point and the LiDAR sensor, there cannot be any other solid object. Due to the height and solid structure of

Algorithm 6: No-Overlap Contractor

Data: Initial set of poses $\mathcal{P}_i = \begin{pmatrix} \mathcal{T}_i & \Phi_i \end{pmatrix}^T$, the building polygons are $\mathcal{B} = \{B_1, ...B_m\}$
Result: Feasible set of poses $\mathcal{P}_i^*$ that do not overlap with the buildings $\mathcal{B}$

Function *NoOverlapContractor($\mathcal{P}_i$,$\mathcal{B}$)*
  $\mathcal{T}_i^* = \varnothing$;
  $S \coloneqq \mathcal{T}_i$;
  while $S \neq \varnothing$ **do**
    $[\mathbf{t}] \coloneqq \mathsf{top}(S)$;
    $\mathcal{B}^* = \mathsf{getCloseBuildings}([\mathbf{t}], \mathcal{B})$;
    for $B : \mathcal{B}^*$ **do**
      $([\mathbf{t}_{\mathsf{in}}], [\mathbf{t}_{\mathsf{out}}]) = \mathcal{S}_{\mathsf{poly}}([\mathbf{t}], B)$;
      if $[\mathbf{t}_{in}] \cap [\mathbf{t}_{out}] = \varnothing$ **then**
        `// Insert to subpaving.`
        $\mathcal{T}_i^* = \mathcal{T}_i^* \cup [\mathbf{t}_{\mathsf{out}}]$;
      else
        if $w([\mathbf{t}_{out}]) > \epsilon$ **then**
          `// Bisect` $[\mathbf{t}_{\mathsf{out}}]$
          $([\mathbf{t}_l], [\mathbf{t}_r]) = \mathsf{bisect}([\mathbf{t}_{\mathsf{out}}])$;
          $S.\mathsf{push}([\mathbf{t}_l])$;
          $S.\mathsf{push}([\mathbf{t}_r])$;
        else
          `// Insert to subpaving.`
          $\mathcal{T}_i^* = \mathcal{T}_i^* \cup [\mathbf{t}_{\mathsf{out}}]$;
        end
      end
    end
  end
  $\mathcal{P}_i^* = \begin{pmatrix} \mathcal{T}_i^* & \Phi_i \end{pmatrix}^T$;
  return $\mathcal{P}_i^*$;

building facades in urban environments, almost all laser beams are blocked – that means the visibility region is restricted – by the building walls. Nevertheless, some exceptions exist: Open doors and non-solid windows allow laser beams to capture measurements beyond the walls. As the modeling depth of the map is limited to the outer geometric structure of the buildings, windows or doors are not represented in the map. Generally, such openings are small compared to the facades, so only a few laser beams shoot through. Consequently, only a few points are captured beyond the facades, which we seek to filter out from the LiDAR point cloud. After applying the filter, the assumption that a local range-and-bearing measurement cannot cross a building wall becomes valid. This assumption provides another constraint, for which we have implemented the so-called *No-Cross Contractor*. This contractor dismisses all infeasible poses that lead to a crossing of a local observation with a building wall on the map.
Please note that this approach does not require the filter to delete all points other than those on the facades. It only needs to delete potential points that lie beyond potential walls reliably. That means points of objects on the street or trees, which are inside the visibility region, satisfy the visibility constraint and represent excellent and reliable information for the No-Cross Contractor.

(a) Orientation Φ_a. (b) Orientation Φ_b. (c) Orientation Φ_c.

Figure 5.7: Subpavings to the sets of feasible poses to three different orientations after the vehicle has moved forward and the No-Overlap Contractor was applied. The orientation interval is visualized in the upper left corner. Figure 5.7b encloses the correct orientation. The robot moves straight forward. The subpavings are updated according to the orientation.

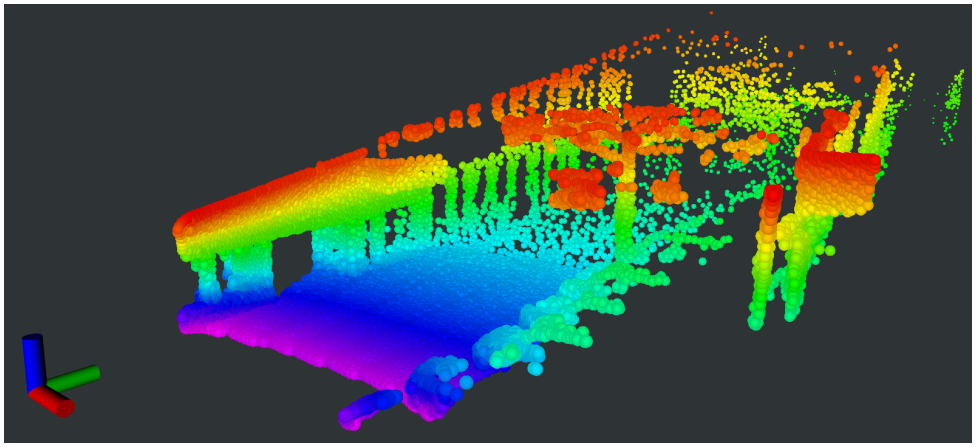


Figure 5.8: Raw LiDAR data. The points are colored based on the z-component.

Filter

An exemplary raw LiDAR scan is depicted in Figure 5.8. The top view of the same scene is depicted in Figure 5.9a. Note that in this exemplary scene, there is an open garage door, which can be well seen in the point cloud depicted in Figure 5.8. The corresponding point cloud in the top view in Figure 5.9a shows that some LiDAR points are also captured inside the garage as the door is open. The goal of the filter is to filter out such structures so that the no-cross assumption becomes valid. For filtering the point cloud, we only consider the mid-points of the LiDAR measurement boxes.

The most restrictive points considering the no-cross constraint are those points that capture the building facades. As illustrated in Figure 5.8, those points usually form clusters on well-defined planes. As we perform the pose estimation in 2D, the facades are represented by points that lie on long lines. As a result, by clustering the points into line segments, we can choose only comparatively long line clusters. The core idea behind this approach is that points captured behind the walls through small openings cannot form large lines such as building facades in the LiDAR scan. By doing this, we can delete potential points behind building

(a) Raw LiDAR data in to view. (b) Virtual 2D scan. (c) Fully filtered.

Figure 5.9: Top view of the LiDAR data at different filtering stages. Figure 5.9a shows the raw data from the solid state LiDAR. The points are colored based on the elevation axis. Figure 5.9b depicts the filtered point cloud (red) after applying the modified virtual 2D scan method [142]. Figure 5.9c presents the results after applying line clustering and filtering. The point cloud is colored blue.

facades as shown in Figure 5.9b. Our filtering approach consists of three steps: First, we project the 3D point cloud to a virtual 2D scan. Second, we cluster the points into lines and select the largest clusters with the highest support. Third, we subsample the clusters to obtain a sparse representation of the building facades.

We first need to project the 3D LiDAR point cloud to 2D to extract line segments. Therefore, we use a similar approach to the virtual 2D scan method in [142]. The virtual 2D scan method first projects all 3D points onto the plane by setting the z-component to zero. Then, the points are sorted into horizontal angle bins. The size of the bins depends on the horizontal angle resolution of the LiDAR sensor. By selecting the point with the largest distance to the LiDAR origin for each angle bin, the authors obtain a 2D boundary map for indoor environments. However, we apply our algorithms in outdoor environments, where we do not always have enclosing boundaries. That is why we modify the boundary point selection. Similar to [142], we also sort the points based on the horizontal angle $\alpha_{\mathsf{h}} = \text{atan2}(y, x)$ using the x- and y-component of the points into horizontal angle bins. We further sort the points within each angle bin based on the horizontal distance $d_{\mathsf{h}} = \sqrt{x^2 + y^2}$ into distance bins. As a result, multiple distance bins are associated with each angle bin, and to each distance bin, multiple points are associated. We determine the vertical span angle of the associated points for each angle and associated distance bin. That means, for each point associated with the distance bin, the vertical angle $\alpha_{\mathsf{v}} = \text{atan2}(\sqrt{x^2 + y^2}, z)$ is determined, and we store the maximum and minimum vertical angles. Hence, for each distance bin, we associate the vertical angle span determined by the difference between the minimal and maximal vertical angles. For each horizontal angle bin, we now select an arbitrary point from the associated distance bin with the largest vertical angle span. This selection algorithm exploits the property that points on facades usually span a large observation vertical angle. Then, we project the selected points to the x-y plane.

By applying this selection algorithm to the LiDAR scan shown in Figure 5.9a, we obtain the modified virtual 2D scan depicted in Figure 5.9b. Up to this point, we managed to reduce the 3D scan to a 2D scan that contains candidate points that may lie on facades. Now we need to

Algorithm 7: Line clustering of ordered points.

Data: Set of points $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$, where $\mathbf{p}_i = \begin{pmatrix} x & y \end{pmatrix}^T$, that are ordered based on the horizontal angle; Maximal line distance d_{max} for association.

Result: Set of line clusters $C = \{L_1, ...L_m\}$, where $L_i = \mathbf{p}_1, ..., \mathbf{p}_k$ is a set of points that correspond to a line.

1 **Function** *LineClustering(P)*
2 $C = \{\}$;
3 **while** $size(P) > 2$ **do**
`// Initialize set of points corresponding to a line.`
4 $L = \{\}$;
`// Retreive neighboring points.`
5 $\mathbf{p}_0 = P[0]$;
6 $\mathbf{p}_1 = P[1]$;
7 L.insert($\mathbf{p}_0$);
8 L.insert($\mathbf{p}_1$);
9 P.erase(0);
10 P.erase(1);
`// Compute the line paramters` $\mathbf{n}$ `and` d.
11 $\mathbf{d} = \mathbf{p}_0 - \mathbf{p}_1$;
12 $\alpha = \text{atan2}(-\mathbf{d}.x, \mathbf{d}.y)$;
13 $\mathbf{n} = \begin{pmatrix} \cos\alpha & \sin\alpha \end{pmatrix}^T$;
14 $d = \mathbf{n}^T \cdot \mathbf{p}_0$;
`// Check line distances to all remaining points.`
15 **for** $i = 0$; $i <$ *size*(P); $i = i + 1$ **do**
16 $\mathbf{p}_i = P[i]$;
17 **if** $|d - \mathbf{n}^T \cdot \mathbf{p}_i| \leq d_{max}$ **then**
18 L.insert($\mathbf{p}_i$);
19 P.erase(i);
20 $i = i - 1$;
21 **end**
22 **end**
23 C.insert(L)
24 **end**
25 **return** C;

cluster the points into line segments. Our clustering algorithm is described in Algorithm 7. The general idea is to determine the line spanned by consecutive pairs of points and to associate points in the vicinity of the line. Therefore, we first need to select consecutive points in the ordered set of points P. In Algorithm 7, the first and second points in the list are selected. Those selected points are erased from the list of non-clustered points. From lines 11 to 14, the 2D line parameters are determined. Then, we associate all the remaining points of the non-clustered list closer than d_{max} with the line. Therefore, we generate a local line list L and delete those points from P. After iterating through all remaining points, we compute the new cluster by repeating the association algorithm.

In Algorithm 7, we exemplarily select the pairs of points that are immediate neighbors to determine a line. Instead, it is also possible to select close points by, for instance, taking every fifth point. This reduces the noise of the line parameters but may lead to neglecting short-line

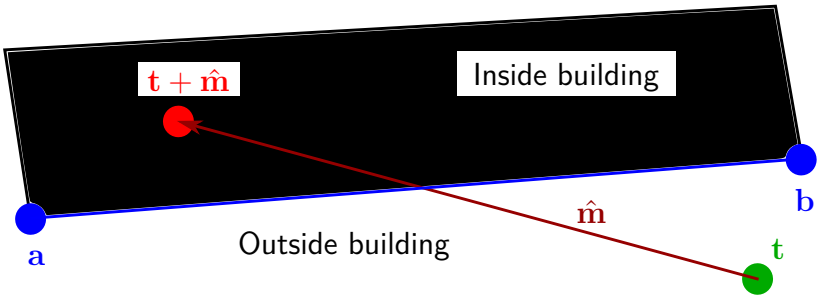


Figure 5.10: No-Cross Contractor variables. The building B is visualized in black. One wall represented by the line segment spanned by $\mathbf{a}$ and $\mathbf{b}$ is highlighted blue. The rotation corrected local measurement is $\hat{\mathbf{m}}$ represented by the red vector.

clusters. As we are interested in comparatively large lines in the 2D virtual scan, skipping a few points to compute the line has proven to be a good choice. In our current implementation, we take every fifth point to determine the local line.

Algorithm 7 provides a set of line clusters from which we only select those that fulfill the following constraints. First, the candidate cluster must contain a minimal number of points. Second, the extracted candidate clusters must exceed a minimum length threshold. The parameters depend on the data and the map topology. For our datasets, we choose a minimal number of 6 points and a minimal length of 7.5 m as those parameters lead to the best results in our empirical studies.

Depending on how well facades are captured by the LiDAR scanner, the density of points may vary in the virtual 2D scan. In the case of very dense points, the No-Cross Contractor does not need to consider all points of a selected cluster to dismiss infeasible parts. To ensure the efficient performance of the No-Cross Contractor, we further subsample from the selected clusters by homogeneously selecting points along the lines. We subsample one point every 1 m along each line. After applying all filtering steps to the LiDAR data depicted in Figure 5.9a, the result is shown in Figure 5.9c.

No-Cross Contractor

The No-Cross Contractor contracts the set of poses to the feasible set, for which each filtered local measurement does not cross a mapped wall. As buildings are represented by polygons, a wall is modeled by a line segment between consecutive corner points. By a blue line, we illustrate an exemplary line segment in Figure 5.10. The corner points $\mathbf{a}$ and $\mathbf{b}$ that span the line are highlighted in blue. The corresponding building is colored black. The contractor is applied to each subset $[\mathbf{t}]$ of the translation subpaving $\mathcal{T}_i$ of the set of poses $\mathcal{P}_i = \begin{pmatrix} \mathcal{T}_i & \Phi_i \end{pmatrix}^T$. First, we must determine the potential line segments that may cross the local measurements. We determine the search radius by adding the length of the half-diagonal of $[\mathbf{t}]$ to the distance of the farthest measured point $\mathbf{p}_{\text{far}}$. The radius search is performed with the KD-Tree. Hence, the radius search provides a list of buildings in the vicinity of $[\mathbf{t}]$.
Now, we apply the No-Cross Contractor for each building and each line segment of the building polygon. Let $\mathcal{L} = \{[\mathbf{a}], [\mathbf{b}]\}$ be a line segment and let $[{}^L\mathbf{m}]$ be a filtered local measurement described in the local LiDAR frame taking the sensor uncertainty into account. For reader

(a) Orientation Φ_a. (b) Orientation Φ_b. (c) Orientation Φ_c.

Figure 5.11: Subpavings to the sets of feasible poses to different orientations after applying the No-Overlap and the No-Cross Contractor. Using the results shown in Figure 5.7 and applying the No-Cross Contractor with the filtered point cloud in Figure 5.9c, the resulting subpavings to the corresponding orientations are shown in the figures.

convenience, we omit the map frame index M. Note that we also consider the map uncertainty by inflating the corner points $\mathbf{a}$ and $\mathbf{b}$ also to boxes. The No-Cross Contractor contracts $[\mathbf{t}]$ such that the observation line spanned by $[{}^L\mathbf{m}]$ does not intersect with the map line segment $\mathcal{L}$. As $[{}^L\mathbf{m}]$ is described in the local LiDAR frame, the observation vector needs to be transformed to the map frame by rotating ${}^L\mathbf{m}$ by φ_i to $\hat{\mathbf{m}}$ for all ${}^L\mathbf{m} \in [{}^L\mathbf{m}]$, $\hat{\mathbf{m}} \in [\hat{\mathbf{m}}]$ and $\varphi_i \in \Phi_i$. By doing this, we preserve the local observation as a local vector but only modify the orientation from the LiDAR frame L to the map coordinate frame M. The no-cross constraint is formulated by

$$\begin{cases} \mathbf{a}_b = \mathbf{b} - \mathbf{a}, \\ \mathbf{a}_m = \hat{\mathbf{m}} + \mathbf{t} - \mathbf{a}, \\ \mathbf{a}_t = \mathbf{t} - \mathbf{a}, \\ \mathbf{t}_a = \mathbf{a} - \mathbf{t}, \\ \mathbf{t}_b = \mathbf{b} - \mathbf{t}, \\ z_1 = \det(\mathbf{a}_b, \mathbf{a}_m) \cdot \det(\mathbf{a}_b, \mathbf{a}_t), \\ z_2 = \det(\hat{\mathbf{m}}, \mathbf{t}_a) \cdot \det(\hat{\mathbf{m}}, \mathbf{t}_b), \\ \max(z_1, z_2) > 0 \end{cases}, \tag{5.3}$$

for all $\mathbf{a} \in [\mathbf{a}]$, $\mathbf{b} \in [\mathbf{b}]$, $\hat{\mathbf{m}} \in [\hat{\mathbf{m}}]$ and $\mathbf{t} \in [\mathbf{t}]$. Figure 5.10 depicts graphically the geometric correspondence between the variables. In the graphical example, (5.3) is not satisfied because $\hat{\mathbf{m}}$ and the line segment between $\mathbf{a}$ and $\mathbf{b}$ intersect. The no-cross constraint is only satisfied if the local observation $\hat{\mathbf{m}}$, does not cross the line segment spanned by $\mathbf{a}$ and $\mathbf{b}$. The No-Cross Contractor implements the constraints in a forward-backward manner [49].

The No-Cross Contractor is applied to each local measurement and each line segment of each building in the vicinity. The intersection over all contracted results replaces the former subset $[\mathbf{t}]$ in the subpaving $\mathcal{T}_i$ of the set of poses $\mathcal{P}_i$.

Applying the No-Cross Contractor to the set of poses in Figure 5.7, the resulting contracted subset with the local observations shown in Figure 5.9c is depicted in Figure 5.11. Note that the no-cross constraint makes discarding large parts of the set of poses possible. The local measurements provide strong constraints on the set of feasible poses. However, the no-cross

(a) Orientation Θ_a. (b) Orientation Θ_b. (c) Orientation Θ_c.

Figure 5.12: Subpavings after applying the GNSS Contractor. Only feasible sets from Figure 5.11 that are inside the circular uncertainty region of the GNSS-measurement are kept.

constraint is only valid if the filter reliably discards measurements that shoot through small openings in the facades. Hence, we should choose the filter parameters conservatively so that the no-cross assumption is always valid. On the downside, the conservative nature of the filter also discards good and restrictive points that will help the No-Cross Contractor to dismiss infeasible parts more efficiently. We refrain from using a q-relaxed intersection as the number of no-cross violations is unknown beforehand and requires further parameter tuning.

5.2.5 GNSS Contractor – Vehicle Inside GNSS-Region

The GNSS Contractor cuts off the region, which is inconsistent with the GNSS-uncertainty region. GNSS can be unreliable in urban environments, so that occasional dropouts may occur. We only apply for the GNSS Contractor if the global position is available. However, the global position can be severely corrupted. Hence, we account for the global positioning error by considering an uncertain region. We define the border of this region by a circle around the global position. The radius of the circle describes the uncertainty. The GNSS Contractor implements a forward-backward contractor [49] for the circle equation

$$(t_x - p_x)^2 + (t_y - p_y)^2 = r^2, \tag{5.4}$$

where $t_x \in [t_x]$ and $t_y \in [t_y]$ is the vehicle position, $p_x \in \mathbb{R}$ and $p_y \in \mathbb{R}$ is the GNSS-based position transformed to the map frame and $r \in [r]$ is the uncertainty radius for the GNSS-based position. We set $[r] = [0, 50]$ in our experiments. Figure 5.12 illustrates the GNSS contractor. Note that large parts of the feasible set from Figure 5.11 are dismissed by the GNSS Contractor. Especially in symmetric scenes where multiple disconnected feasible solutions can co-exist, the GNSS data provides precious information, although it can be comparatively uncertain.

5.2.6 Rotation bisection

The introduced contractors are only applied on the translation subpaving of each set of poses $\mathcal{P}_{i,t}$ (rotation bin) where each $\mathcal{P}_{i,t}$ is processed by an independent thread in parallel as shown in Figure 5.13. Each sub-figure in Figure 5.13 represents a rotation bin. The according orientation

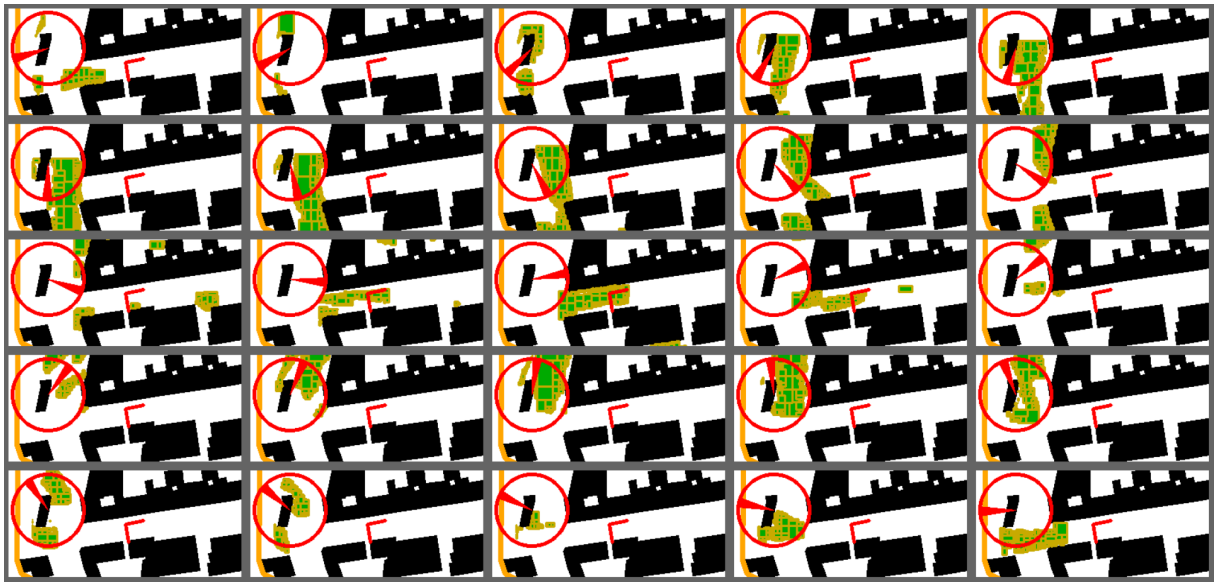
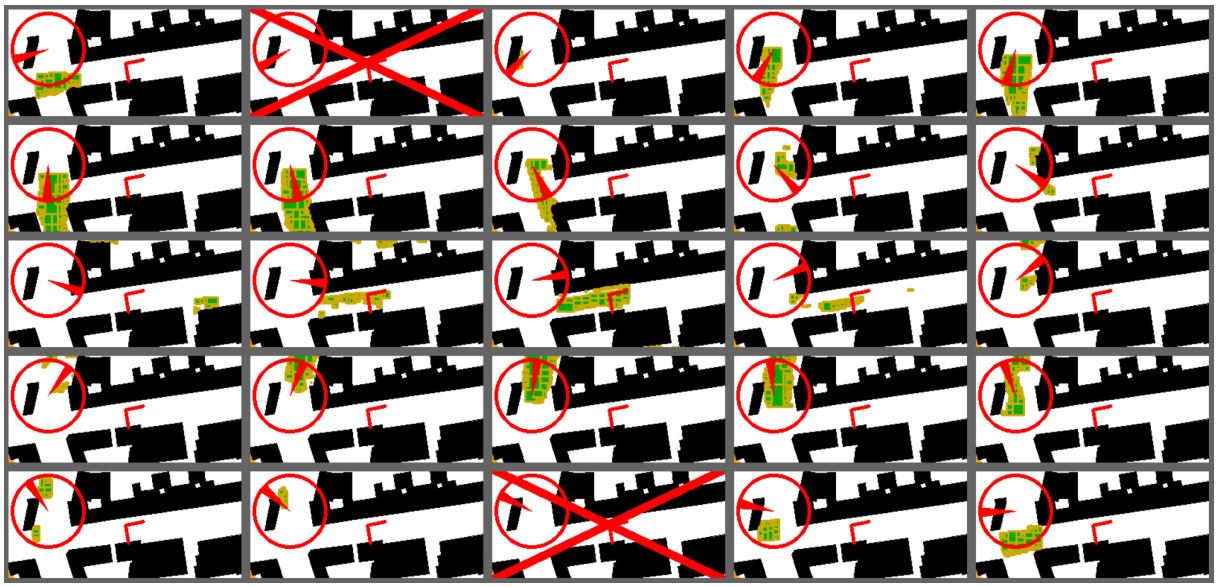


Figure 5.13: Rotation bins. The contractors are applied to each rotation bin independently. The corresponding rotation interval to the bin is visualized in the top left corner. The feasible set of translations is shown in green. In total, 25 rotation bins are considered here.

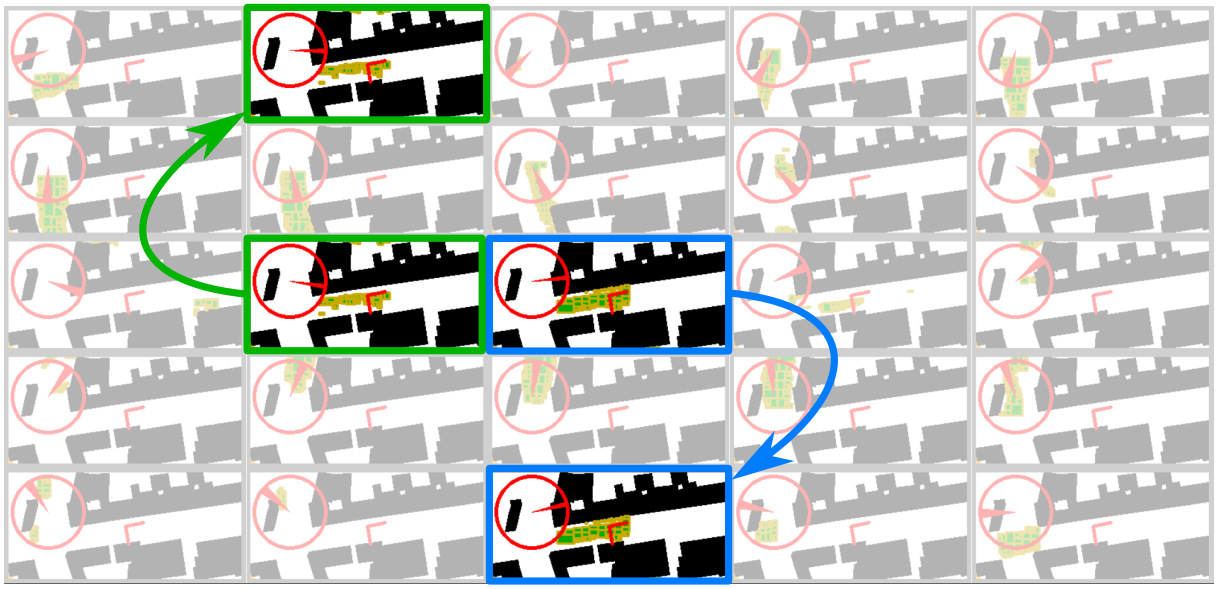
set $\Phi_{i,t}$ is illustrated in the top left corner, and the set of translations $\mathcal{T}_{i,t}$ is colored green. The corresponding set of rotations $\Phi_{i,t}$ is not directly contracted. Up to now, we only considered the passive contraction of the feasible set of rotations for the case when the $\mathcal{T}_{i,t}$ of $\mathcal{P}_{i,t}$ becomes empty. In this case, we can dismiss $\Phi_{i,t}$ from the feasible set of orientations as illustrated in Figure 5.14a. However, uncertain odometry data provided by our visual odometry module also let the set of orientations $\Phi_{i,t}$ for each bin grow. We consider further bisection of the rotation parameter to improve the passive rotation contraction.

If a rotation bin $\mathcal{P}_{i,t}$ is identified as infeasible, since the translation $\mathcal{T}_{i,t}$ became empty (cf. Figure 5.14a), the workload on the CPU decreases since one thread terminates. We use the freed thread to process a new rotation bin. Therefore, we select the set of poses $\mathcal{P}_{j,t}$ that has the largest translation subpaving $\mathcal{T}_{j,t}$. We perform the rotation splitting by bisecting $\Phi_{j,t}$ to $\Phi_{j_l,t}$ and $\Phi_{j_r,t}$. Each of the obtained rotation intervals forms a new rotation bin. Hence, from $\mathcal{P}_{j,t} = \begin{pmatrix}\mathcal{T}_{j,t} & \Phi_{j,t}\end{pmatrix}^T$ we obtain $\mathcal{P}_{j_l,t} = \begin{pmatrix}\mathcal{T}_{j,t} & \Phi_{j_l,t}\end{pmatrix}^T$ and $\mathcal{P}_{j_r,t} = \begin{pmatrix}\mathcal{T}_{j,t} & \Phi_{j_r,t}\end{pmatrix}^T$. Note that the translation parts of $\mathcal{P}_{j,t}$, $\mathcal{P}_{j_l,t}$ and $\mathcal{P}_{j_r,t}$ are identical since we simply copy the translation subpaving $\mathcal{T}_{j,t}$ from the initial set $\mathcal{P}_{j,t}$. Only the rotation parameters are different. While we exchange $\mathcal{P}_{i,t}$ by $\mathcal{P}_{j_l,t}$ in the ith thread, the jth thread continues the process on the right child $\mathcal{P}_{j_r,t}$ of the bisection. Figure 5.14b illustrates the rotation splitting.

While the rotation bin size can be chosen initially coarsely, the bisection narrows down the feasible rotation by decoupling rotation subsets. Since this is done dynamically based on the available computational resources, the workload is well-balanced along the trajectory. Note that we only perform the rotation bisection if the orientation interval $\Phi_{j,t}$ exceeds a defined threshold to avoid unnecessary bisections. This means that if the orientation interval is comparatively small, we do not occupy the complete computational resources of the CPU.



(a) Detection of infeasible rotation bins. The corresponding rotation interval is considered infeasible if the set of translations becomes empty for a rotation bin. Hence the feasible set of rotations is contracted. In this case, two rotation bins are detected as infeasible, indicated by a red cross.



(b) Bisection of the largest rotations bins. As infeasible rotation bins are detected as shown in Figure 5.14a, the rotation bins are replaced by the right child of the bisection of the rotation bins with the largest translation subset. Note that the rotation intervals of the obtained rotation bins shrink. In this case, two rotation bins are replaced based on the bisection of the rotation bins with the largest translation subpavings.

Figure 5.14: Bisection of the rotation intervals. As rotation bins are associated with threads, rotation bins that are detected as infeasible are replaced by bisected rotation bins to balance the workload and obtain optimal parallelism.

(a) Feasible set. (b) Spread particles. (c) Resample.

Figure 5.15: Bounded Monte Carlo Localization. Figure 5.15a shows the set of feasible poses in green. The image shows a snapshot of a trajectory. The red pose is the ground truth vehicle pose for the snapshot. The red line visualizes the driven trajectory. Figure 5.15b shows the randomly spread particles inside the feasible set. Blue coordinate frames represent the particles. The aggressive resampling only permits particles with very high weights, while other particles get omitted in this step as visualized in Figure 5.15c

5.3 Bounded Monte Carlo Localization

The interval-based localization provides a feasible set of poses for the robot. This section focuses on the most likely solutions within the feasible set. That means the feasible set can be considered the exploration region where we seek to find the most consistent solution by applying a modified MCL approach. Classical MCL approaches do not track the feasible solution. That is why those methods need to keep track of a large variety of particles along the trajectory so that they do not lose particles that may be good candidates. That implies that the classical approach needs many particles. Since in our hybrid approach the interval method already provides the feasible solution set, we only need a few particles in combination with an aggressive resampling procedure in our proposed bounded MCL.

To efficiently spread the particles, we consider the rotation bins of the interval method. Each set of poses $\mathcal{P}_i$ represents a rotation bin and the corresponding feasible position of the vehicle. That is why we allocate for each $\mathcal{P}_i$ a specified number of particles that should be spread within the feasible solution set. In our experiments, we set the maximal number of particles for one subset $\mathcal{P}_i$ to 20. This is illustrated Figure 5.15. While Figure 5.15a displays all feasible sets of poses determined by the interval-based approach, Figure 5.15b shows the randomly spread particles all over the feasible set.

For each time step, each particle is updated by the odometry as explained in Subsection 5.2.2. The update also represents a random process where we sample from a uniform error distribution. The update step may cause particles to leave the feasible solution set. In that case, we delete those particles, as the represented solution is not considered feasible. For each deleted particle

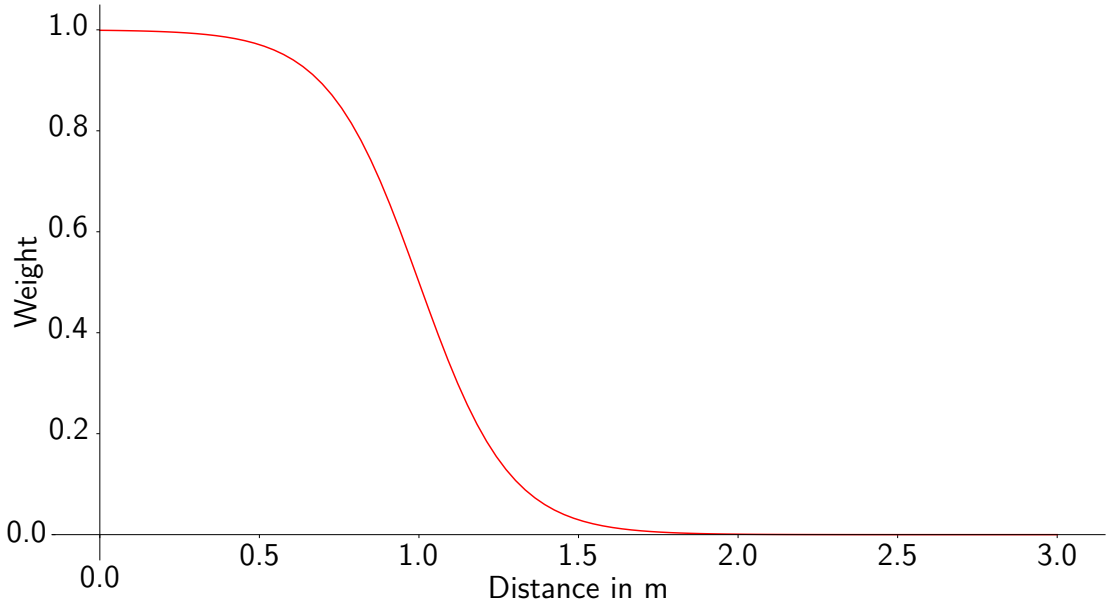


Figure 5.16: Mapping function from distance to weight.

associated with a rotation bin $\mathcal{P}_i$, we randomly spread a new particle inside the feasible solution set in $\mathcal{P}_i$. This enables us to check for better solutions in the feasible set.

We weight the particles based on the virtual 2D scan extracted from the LiDAR data as introduced in Subsection 5.2.4. Therefore, using the pose of the particles, the local LiDAR points are transformed from the local frame to the map frame. For each transformed LiDAR point, we determine the distance to the nearest facade. Therefore, we employ the KD-Tree for a nearest-neighbor search. The particles are weighted based on the transformed points' proximity to the nearest facade. To map the distance of a transformed point to a weight, we choose the modified sigmoid function

$$f_{\text{weight}}(d) = 1 - \frac{1}{1 + \exp(-7 \cdot (|d| - 1))}. \tag{5.5}$$

The variable d defines the orthogonal distance of the transformed point to the closest facade. In Figure 5.16, the mapping function is plotted. As shown in Figure 5.16, the closer a point to a facade, the higher the weight of this local measurement. If the local measurement has a distance of more than 2 m, it is weighted zero. We define the particle weights as the sum of all local measurement weights. If all transformed local measurements fit well to the facades, the particle's weight will be the number of all local measurement points.

We perform a weighted resampling in each iteration step to replace particles with low weights. High-weighted particles are directly chosen for the next iteration step and are unaffected by the resampling process. Particles are classified as highly weighted if their weight is at least $90\,\%$ of the maximal weight among all particles in the current iteration step. All those particles that have lower weights are resampled from the high-weighted particles so that all low-weight particles are replaced. In the resampling process, we randomly manipulate the pose of the particle so that we do not precisely resample a copy of the particle but retrieve a similar one.

Hence we have two sources of producing new particles: First, as explained above, for each deleted particle – since they exited the feasible set – a new particle is randomly generated in

the feasible region. This makes exploring regions inside the feasible set that are not covered by particles possible. Second, we generate new particles from the high-weighted particles by applying minor random manipulations on the pose of the particles. The odometry uncertainty determines how large the manipulations are. This enables a local exploration of the region in the vicinity of high-weighted particles, making convergence to the correct pose possible. Figure 5.15c shows the particles inside the feasible set after applying the aggressive resampling approach on the particles shown in Figure 5.15b. Note that only the few best particles can survive this resampling procedure.

The combination of deleting infeasible and randomly spreading new particles leads to fast convergence of the bounded MCL to the most likely solution. Note that this resampling procedure is only possible because the interval method provides the feasible set of poses at each time step so that particles are only spread in those feasible regions. We do not depend on a large set of different particles since, in the case of particle depletion, the interval-based approach provides us with the region where new particles can be spread. Hence, in the case of relocalization, we do not have to spread particles all over the map again as classical MCL does. Instead, we can concentrate on the feasible region by spreading only in those parts, increasing the convergence speed.

5.4 Module Output

The first module in our pipeline is the visual odometry. The module provides the relative motion of the vehicle described in a defined reference frame. However, the odometry approach suffers from drift as we only consider local information. That is why we have to incorporate global information into the pose estimation. Buildings in urban environments represent distinguishable and detectable landmarks that we can use for localization. Additionally, building maps are publicly available, making the localization approach globally applicable. Nonetheless, as the visual odometry only provides the relative motion of the vehicle described in a local reference frame, we do not know where the vehicle is located initially. Especially in urban canyons, where GNSS measurements are very inaccurate due to multipath effects and blockage, the initial position uncertainty can be very high. In this chapter, we have presented a hybrid localization method that can cope with such large uncertainties and provides, on the one hand, a feasible set of poses that satisfy simple but always valid constraints and, on the other hand, most likely poses within the feasible set represented by particles. As a result, the method presented in this chapter solves the localization problem under large uncertainties, providing a first coarse estimate of the vehicle's pose.

Nonetheless, the feasible set can be comparatively large since only very pessimistic but always valid constraints determine the feasible set of poses. However, by exploiting the best particle estimate on the vehicle pose, we can improve the localization result: Up to now, we did not consider for the set-membership-based approach the direct association of LiDAR points to building facades. This becomes feasible now since the coarse localization already provides an acceptable initial estimate of the vehicle's pose. The association-based refinement of the vehicle pose in the building map is introduced in the next chapter, which completes our HyPaSCoRe Localization pipeline.

6 Refined Localization

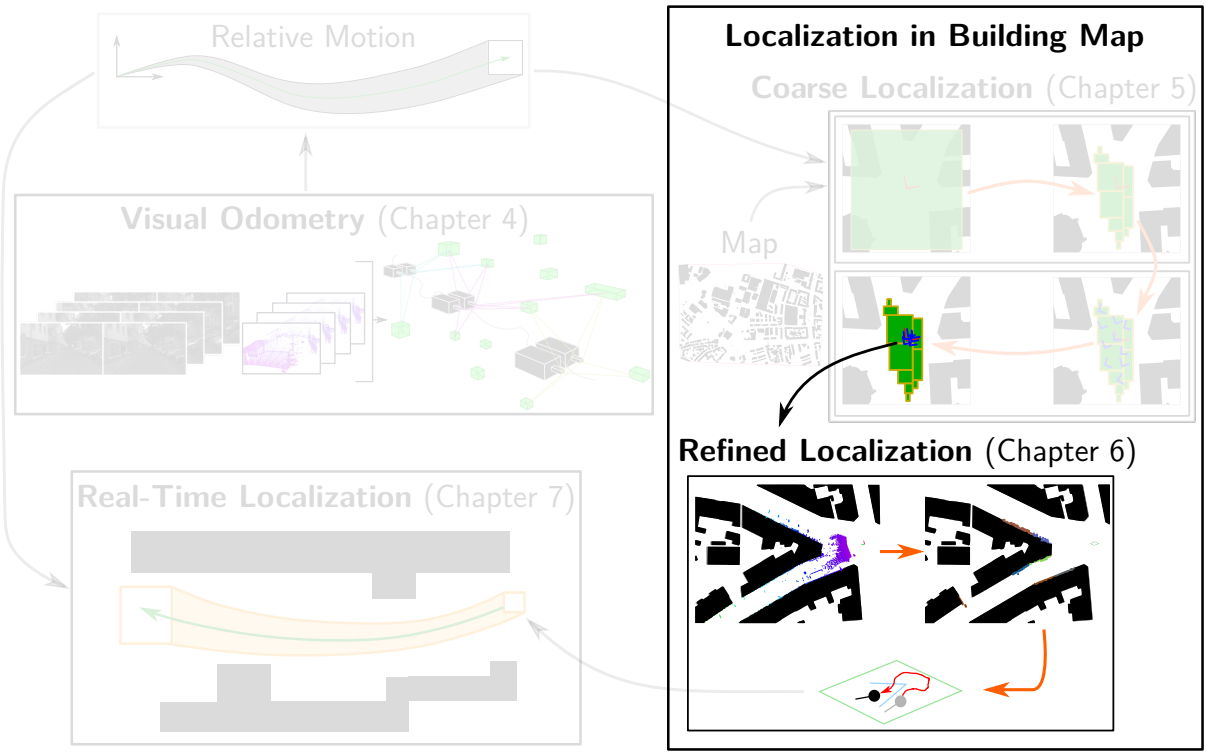


Figure 6.1: The refined localization is the third module in the HyPaSCoRe Localization pipeline and refines the localization estimates provided by the coarse localization module.

The previous chapter introduced the coarse localization that uses the ego-motion information from the visual odometry and the building map to localize the vehicle. However, the estimation of the coarse localization module can be very uncertain, as the feasible region is only contracted on very pessimistic but always valid constraints. Furthermore, the aggressive resampling method in the bounded MCL provides fast convergence. However, it suffers from error-prone tracking: The method can be rapidly attracted by alternative solutions inside the feasible set if the observation model does not provide a well-distinguishable particle weighting. Since the vehicle is equipped with LiDAR sensors that provide measurements with centimeter accuracy, the coarse localization does not fully exploit the potential of the LiDAR sensor. In this chapter we bridge this gap and provide a stable localization estimate.

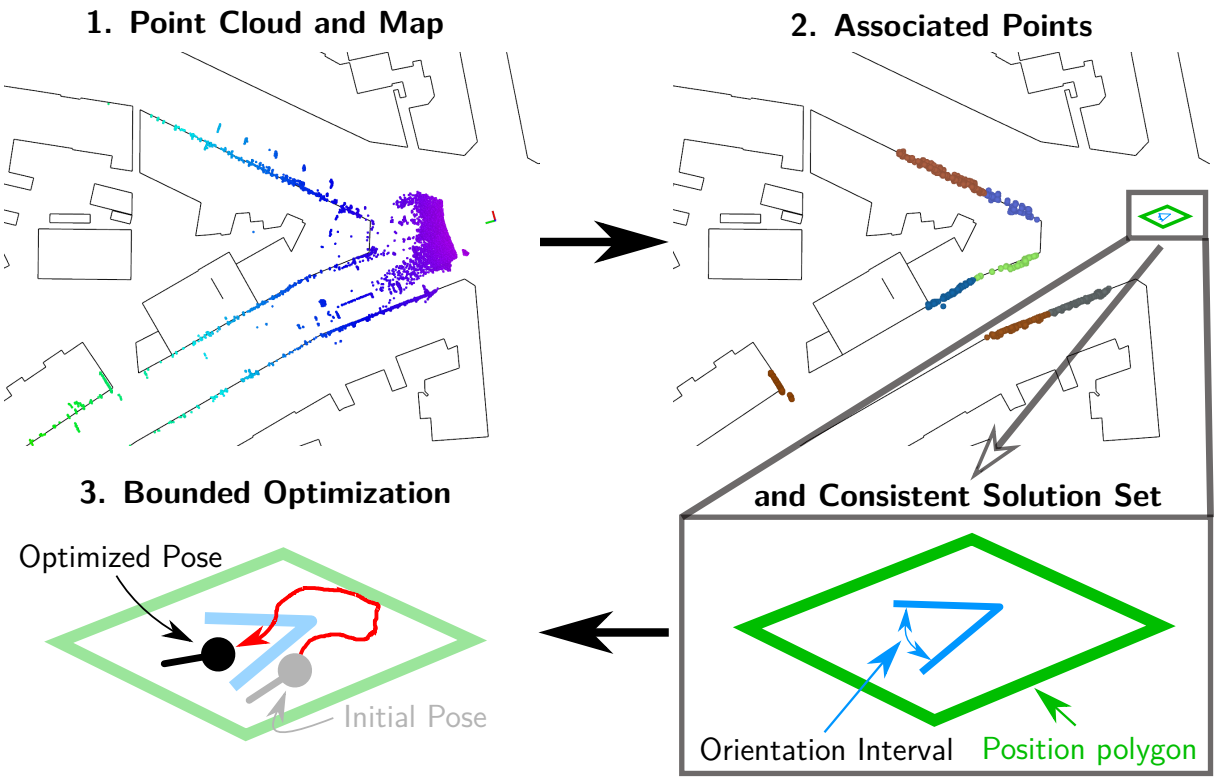


Figure 6.2: Overview. From the point cloud, lines are extracted and associated with the facades on the map. By applying interval analysis, we narrow down the consistent set of poses based on the association. The set of positions is described by a polygon (green), and the orientation (blue) by an interval. We find the most likely solution within the consistent set using a bounded optimization method.

We have not considered a point-to-facade association of LiDAR measurements for the interval-based localization. The association was not possible until now because at least a rough estimate of the vehicle's location was needed. Since the previous chapter provided the tools to solve the coarse localization and particles inside the feasible set indicate which pose is the most likely one, we can perform the data association that will provide further constraints on the location, significantly reducing the pessimism as shown in the localization pipeline overview in Figure 6.1.

This chapter presents a real-time capable hybrid interval-probabilistic refined localization method that improves the coarse localization results. The overview of our method is depicted in Figure 6.2. The fundamental difference between coarse and refined localization is the knowledge about the initial pose. While for the refined localization, the initial pose is known with a small uncertainty, for the coarse localization, the initial pose information is very uncertain or even unknown. This chapter assumes that the initial pose is known from the coarse localization. While association-based localization provides more accurate localization results than the coarse localization, it comes with the cost of stronger assumptions: If we associate a point to a facade in the map, we need to consider geometric assumptions that allow us to suppress alternative constellations. Hence, we cannot guarantee that a considered association is correct. As a result, we cannot guarantee that the localization result will be correct – we only know that it is consistent to the association. That means, in this chapter, we gain more accuracy for our

localization result with the cost of less integrity. The obtained solution is evaluated in the HyPaSCoRe Localization regarding its reliability. If the reliability is high, the consistent set provided by the refined localization is used to contract the pessimistic feasible set from the coarse localization.

Our hybrid approach consists of two steps. First, the interval-based approach narrows down the orientation to a smaller interval. It provides a set described by a minimal polygon for the vehicle's position that encloses the consistent set of poses by associating local LiDAR points to building facades. Second, we perform a probabilistic MLE to determine the most likely solution within the determined consistent set. The MLE is converted into a least-squares problem that is solved by an optimization approach that considers the bounds of the solution set so that only a solution within the consistent set is selected as the most likely one.

The approach has already been published in [143]. In the scope of this work, we provide further insights into our approach and present a more detailed description of our method. Furthermore, this work aims to embed the method presented in [143] in the entire HyPaSCoRe Localization pipeline.

6.1 Task Description, Notations and Assumptions

The initial pose of the robot is known up to a defined uncertainty. The goal of our hybrid interval-probabilistic method is to provide the set of all consistent poses ${}^{M}\mathcal{P}_{L_t}^{\triangle}$ and the most likely pose ${}^{M}\hat{\mathbf{T}}_{L_t}^{*} \in {}^{M}\ \mathcal{P}_{L_t}^{\triangle}$ of the vehicle for each time step in a given map that consists of building footprints using 3D LiDAR data. We define a single pose ${}^{M}\hat{\mathbf{T}}_{L_t} = \begin{pmatrix} t_x & t_y & \varphi \end{pmatrix}^T$ by two translation parameters and one orientation parameter in the 2D space. Accordingly, a set of poses ${}^{M}\mathcal{P}_{L_t}^{\triangle} = \{{}^{M}\mathcal{T}_{L_t}^{\triangle}, [{}^{M}\varphi_{L_t}]\}$ is described by a polygon ${}^{M}\mathcal{T}_{L_t}^{\triangle}$ that defines the translation and an interval $[{}^{M}\varphi_{L_t}]$ that defines the set of orientations. We describe a polygon ${}^{M}\mathcal{T}_{L_t}^{\triangle}$ by its border defined by an ordered set of corner points $\mathcal{C} = \{{}^{M}\mathbf{c}_1, \ldots, {}^{M}\mathbf{c}_n\}$ with ${}^{M}\mathbf{c}_i = \begin{pmatrix} x & y \end{pmatrix}^T$ for $n, i \in \mathbb{N}$ and $x, y \in \mathbb{R}$. Hence, the border of ${}^{M}\mathcal{T}_{L_t}^{\triangle}$ is determined by the line segments between corner points. We define a position $\begin{pmatrix} t_x & t_y \end{pmatrix}^T \in {}^{M}\ \mathcal{T}_{L_t}^{\triangle}$ by a point that lies within or on the border of ${}^{M}\mathcal{T}_{L_t}^{\triangle}$. Note that the polygonal representation of the feasible set of positions is a generalization of the classical interval-based 2D box representation. That means, if the polygon simplifies to an axis-oriented rectangle ${}^{M}\mathcal{T}'_{L_t}$, the polygon is well represented by a box that consists of two intervals $\begin{pmatrix} [t_x] & [t_y] \end{pmatrix}^T \subseteq {}^{M}\ \mathcal{T}'_{L_t}$. As the translation part of ${}^{M}\mathcal{P}_{L_t}^{\triangle}$ always refers to a polygonal set representation, in the scope of this chapter will omit the polygon qualifier $\triangle$, and we will write ${}^{M}\mathcal{P}_{L_t}$.

The initial pose of the robot is known for a defined uncertainty. This uncertainty cannot be arbitrarily large. As the proposed method in this chapter cannot cope with ambiguous localization results, we assume the initial pose uncertainty to be large – but small enough to unambiguously match local data to map facades. Further, without loosing generality we assume the initial position ${}^{M}\mathcal{T}_{L_0} \supseteq \begin{pmatrix} [t_{x,0}] & [t_{y,0}] \end{pmatrix}^T$ to be a box. Hence, we approximate the feasible set of positions from the coarse localization by enclosing it with an axis-aligned box.

Given the map and the LiDAR data, the goal of our interval approach is to determine a minimal position polygon ${}^{M}\mathcal{T}_{L_t}$ and the orientation interval $[{}^{M}\varphi_{L_t}]$ that enclose all vehicle

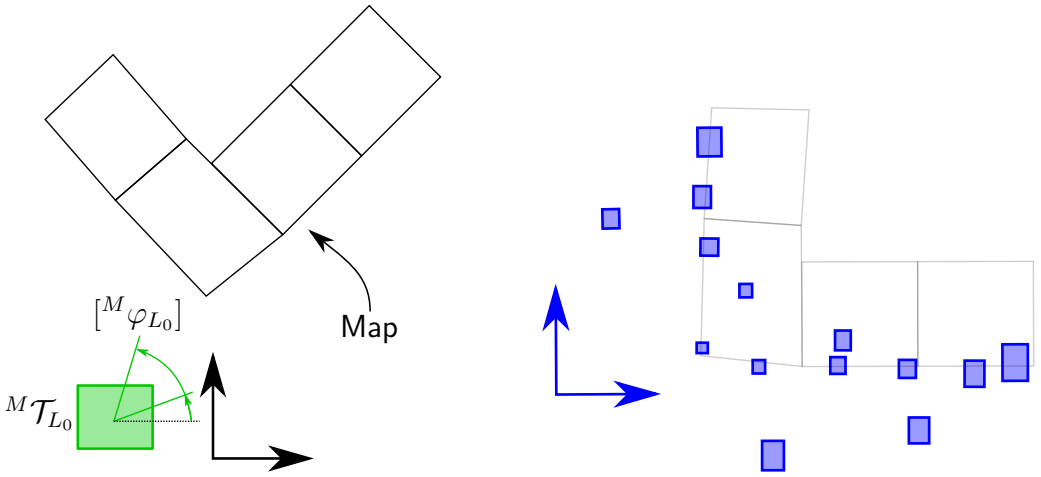
poses that are consistent with the point-to-facade associations. We define a polygon as minimal concerning a set $\mathcal{S}$ if no other polygon has a smaller area for a finite number of corners. The bounded MLE aims to find the most likely pose among the consistent set of poses determined by the interval-based method.

6.2 Interval-based Refinement

The goal is to refine the localization estimate from the coarse localization. Therefore, we need to adapt the available data accordingly. Since the LiDAR provides 3D boxes as measurements under consideration of an interval-based sensor error model, we first need to project the boxes to 2D by omitting the elevation axis.
The interval-based refinement approach consists of five steps. First, the previous localization estimate needs to be updated by the vehicle motion provided by the visual odometry module as presented in Subsection 6.2.1. Second, we need to determine which of the building facades are seen. Therefore, we introduce the map line selection in Subsection 6.2.2, which determines the association between local LiDAR measurements and the building facades on the map. That means clusters of local measurements are associated with the facades in the vicinity. However, the association does not distinguish between facade and non-facade measurements. Hence, the association step may also associate, for instance, points that lie on balconies, ornaments, or close objects that are seen in the LiDAR sensor. The goal of the third step – the interval-based Hough Transformation (iHT) – is to extract a line with the highest support from each of the clusters of projected LiDAR boxes that correspond to the associated facade. The iHT is presented in Subsection 6.2.3. Based on the extracted local line parameters and the associated facades in the map, in the fourth step presented in Subsection 6.2.4, we contract the orientation interval of the pose to the set of consistent orientations under consideration of the association. In the fifth step in Subsection 6.2.5, we determine the minimal position polygon by exploiting the detection accuracy of the facades in the local data and considering the building geometry. For reader convenience, we introduce an exemplary scenario in Figure 6.3 to visually describe the steps in our approach. The exemplary building map is shown in Figure 6.3a. The map frame M is colored black. The initial pose parameters are visualized in green. As defined above, the initial position is defined by a box ${}^{M}\mathcal{T}_{L_0}$ and the orientation interval $[{}^{M}\varphi_{L_0}]$ is represented by a set of headings. In Figure 6.3b, the local LiDAR measurement boxes are illustrated in blue. The local LiDAR coordinate frame L_t is also colored blue.

6.2.1 Odometry Update

Suppose the refined localization has not been initialized. In that case, we assume an initial position ${}^{M}\mathcal{T}_{L_0}$ described by an interval box and the initial orientation $[{}^{M}\varphi_{L_0}]$ described by an interval to be given. In our HyPaSCoRe Localization system, the coarse localization provides this initial estimate. However, if the localization has already been initialized, the polygon ${}^{M}\mathcal{T}_{L_{t-1}}$ and the orientation interval $[{}^{M}\varphi_{L_{t-1}}]$ from the previous step are available. To obtain an initial estimate on the set of poses for time t based on the previous set $t-1$, we predict



(a) Map and initial pose. (b) Local LiDAR measurement boxes.

Figure 6.3: Exemplary scenario. Figure 6.3a shows a map and the initial pose estimate. The map components are colored black. Fig. 6.3b shows the local measurements, which are colored blue.

the vehicle pose using odometry data. Therefore we use the interval-based odometry estimate from the visual odometry module.

To predict the orientation interval, the same approach as presented in Subsection 5.1.1 by simply adding the relative orientation $\Delta\varphi$ to the orientation at time $t-1$ can be used:

$$^M\varphi_{L_t} =^M \varphi_{L_{t-1}} + \Delta\varphi. \tag{6.1}$$

Note, that $^M\varphi_{L_{t-1}} \in [^M\varphi_{L_{t-1}}]$, $\Delta\varphi \in [\Delta\varphi]$ and $^M\varphi_{L_t} \in [^M\varphi_{L_t}]$ holds. Since we use the interval extended addition operator, we obtain the predicted orientation interval $[^M\varphi_{L_t}]$. Since $[\Delta\varphi]$ has a non-zero width, the orientation interval will inflate by the odometry uncertainty.

While the rotation update is similar to the approach presented in Subsection 5.1.1, the translation update is differently handled since we represent the translation $^M\mathcal{T}_{L_{t-1}}$ by a polygon instead of a subpaving as we do in the coarse localization. Hence, at time $t-1$ the set of positions $^M\mathcal{T}_{L_{t-1}}$ is represented by a polygon with the corner points $\mathcal{C} = \{^M\mathbf{c}_1, \ldots, ^M\mathbf{c}_n\}$. To obtain the predicted set of poses for time t we transform the set $^M\mathcal{T}_{L_{t-1}}$ by the odometry estimate $[\Delta\varphi]$ and $[\Delta\mathbf{t}]$. Unfortunately, we cannot directly apply interval tools to perform the transformation as $^M\mathcal{T}_{L_{t-1}}$ is defined by points that define the borders of the set. One possibility is to use subpavings as introduced in the coarse localization. However, this involves more computation as we approximate the polygon by a set of smaller boxes. That is why we approximate the polygon $^M\mathcal{T}_{L_{t-1}}$ by an axis-aligned box hull that encloses the polygon. This enables the direct application of interval tools, with the cost of introducing further pessimism due to the wrapping effect. Hence, we can use the same odometry update equation

$$^M\mathbf{t}_{L_t} =^M \mathbf{t}_{L_{t-1}} + \begin{pmatrix} \cos{}^M\varphi_{L_{t-1}} & -\sin{}^M\varphi_{L_{t-1}} \\ \sin{}^M\varphi_{L_{t-1}} & \cos{}^M\varphi_{L_{t-1}} \end{pmatrix} \cdot \Delta\mathbf{t} \tag{6.2}$$

as presented in 5.1.1. Since all involved variables are described in the interval domain, we obtain the predicted set of positions $^M\mathcal{T}_{L_t}$ described by an interval box.

6.2.2 Map Line Selection

In this step, we determine which local measurements can potentially be associated with the facades. We consider two different approaches for the association. First, we introduce the set-membership-based approach in Part 6.2.2.1. When the feasible set provided by the coarse localization is small enough to make unambiguous matching possible, we can transform the local measurements to the map frame using the feasible solution set. This inflates the local measurements in the map frame. Based on the intersection, we can perform the association.

However, if the feasible solution set is comparatively large, the transformed measurements will be large boxes. Hence, the boxes may intersect with multiple facades, so an unambiguous association is impossible. We introduce the second method in Part 6.2.2.2 that represents a typical probabilistic association. Therefore, the bounded MCL in the coarse localization provides valuable probabilistic information: Although the feasible set may be large, particles may form clusters in the most likely regions. If a particle cluster stably tracks the vehicle pose and the particle weights are large for a comparatively large time interval, the particles are likely to be close to the correct pose. Hence, we can use the particles with the highest weight to perform the association.

6.2.2.1 Set-Memebership-based Association

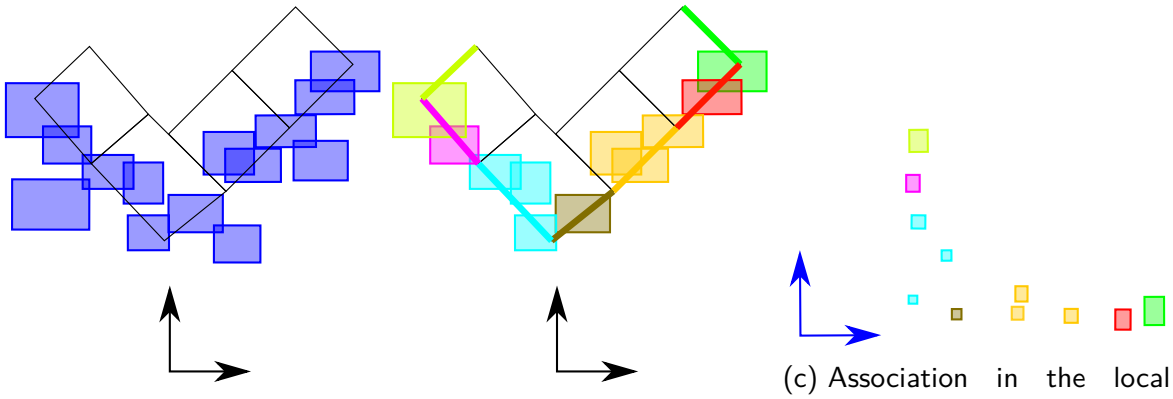
To associate local LiDAR measurements with building facades, we first transform the local measurements to the map frame by using the predicted estimate $[{}^{M}\varphi_{L_t}]$ and ${}^{M}\mathcal{T}_{L_t}$. Hence, the transformation of a local measurement box $[{}^{L}\mathbf{m}]$ to measurement $[{}^{M}\mathbf{m}]$ described in the map frame is defined by

$$ {}^{M}\mathbf{m} = \begin{pmatrix} \cos({}^{M}\varphi_{L_t}) & -\sin({}^{M}\varphi_{L_t}) \\ \sin({}^{M}\varphi_{L_t}) & \cos({}^{M}\varphi_{L_t}) \end{pmatrix} \cdot {}^{L}\mathbf{m} + \begin{pmatrix} t_{x,t} \\ t_{y,t} \end{pmatrix}, \tag{6.3} $$

for ${}^{M}\mathbf{m} \in [{}^{M}\mathbf{m}]$, ${}^{M}\varphi_{L_t} \in [{}^{M}\varphi_{L_t}]$, ${}^{L}\mathbf{m} \in [{}^{L}\mathbf{m}]$ and $\begin{pmatrix} t_{x,t} & t_{y,t} \end{pmatrix} \in {}^{M}\mathcal{T}_{L_t}$. We determine $[{}^{M}\mathbf{m}]$ by applying interval operations in (6.3). As $[{}^{M}\varphi_{L_t}]$ and ${}^{M}\mathcal{T}_{L_t}$ are uncertain, the interval operations propagate the uncertainty to the estimate of $[{}^{M}\mathbf{m}]$, so that the measurement boxes inflate in the transformation step. For the scenario depicted in Figure 6.3, the result of the transformation is illustrated in Figure 6.4a.

The transformation of the local measurements $[{}^{L}\mathbf{m}]$ to the map frame enables us to associate the local measurements to the map facades by simply checking for intersections: If a transformed measurement $[{}^{M}\mathbf{m}]$ has a non-empty intersection with a facade in the map, we can deduce that under the consideration of the initial pose uncertainties and LiDAR measurement uncertainties that $[{}^{M}\mathbf{m}]$ can be a potential measurement on the facade. To identify if a box has a non-empty intersection, we consider two constraints that must be fulfilled. For a given facade $F_i = \{[{}^{M}\mathbf{a}_1], [{}^{m}\mathbf{a}_2], [{}^{M}\alpha], [{}^{M}d]\}$ a box $[{}^{M}\mathbf{m}]$ that satisfies both constraints

$$ \begin{cases} \text{i) } \begin{pmatrix} \cos([{}^{M}\alpha]) & \sin([{}^{M}\alpha]) \end{pmatrix} \cdot [{}^{M}\mathbf{m}] \cap [{}^{M}d] \neq \varnothing \\ \text{ii) } ([{}^{M}\mathbf{a}_1] \sqcup [{}^{m}\mathbf{a}_2]) \cap [{}^{M}\mathbf{m}] \neq \varnothing \end{cases}, \tag{6.4} $$



(a) Transform to map frame. (b) Intersection with F_i. (c) Association in the local frame.

Figure 6.4: As shown in Figure 6.4a, the boxes are inflated by the transformation from L to M due to the predicted pose uncertainty. In Figure 6.4b, those boxes with a non-empty intersection with a facade are associated with it. Based on the association, Figure 6.4c shows the associated local box clusters.

has a non-empty intersection with F_i. The first constraint checks if $[{}^M\mathbf{m}]$ has a non-empty intersection with the line spanned by F_i. Further, the second constraint validates if $[{}^M\mathbf{m}]$ is within the line segment defined by the end and start point of F_i. If a measurement intersects with a facade, we qualify the measurement as a potential facade observation. The presented idea is very similar to the set-membership-based data association presented in [144] which considers the constellation of the locally observed landmarks and the map. To ensure efficient intersection checks, we only consider facades F_i that are in the vicinity of $[{}^M\mathbf{m}]$. Therefore we perform a radius search in the map KD-Tree, where the mid of $[{}^M\mathbf{m}]$ determines the center and the half diagonal length of $[{}^M\mathbf{m}]$ the radius for the radius search. In Figure 6.4b, the intersection-based association of the transformed boxes from Figure 6.4a is visualized. Measurements that are associated with a particular facade are accordingly colored. Note that those local measurements that do not intersect with any of the facades can be discarded since those measurements cannot be potential measurements on the facade. Furthermore, boxes that have intersections with multiple facades can also be associated with each of them. This is not considered in Figure 6.4b for the sake of simplicity. Based on the association in the map frame as illustrated in Figure 6.4b, we also know which measurement in the local frame is associated with which facade in the map (see Figure 6.4c). As a result, the map line selection segments the local measurement boxes into clusters associated with seen facades on the map. This set-membership-based association approach was presented and evaluated in [143].

6.2.2.2 Probabilistic Association

The set-membership-based association considers the uncertainty of the current pose estimate to perform the association of local LiDAR points to building facades. While this approach provides reliable results, large uncertainties may hinder unambiguous association, making the approach inapplicable. However, the bounded MCL in the coarse localization provides the most likely poses represented by particles which we can use for the association. However, we cannot guarantee that the most likely poses are close to the correct pose. To analyze if a particle can be considered for the association, we consider the temporal stability of the particle. Therefore,

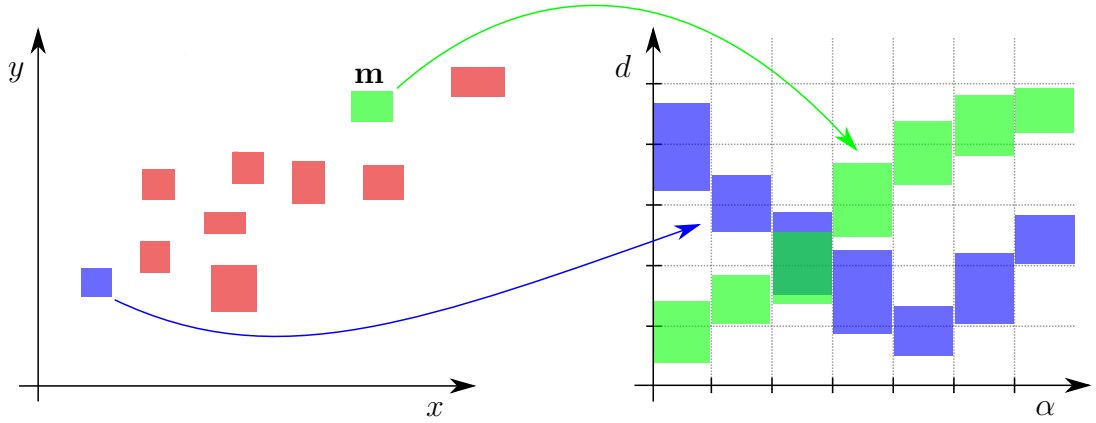


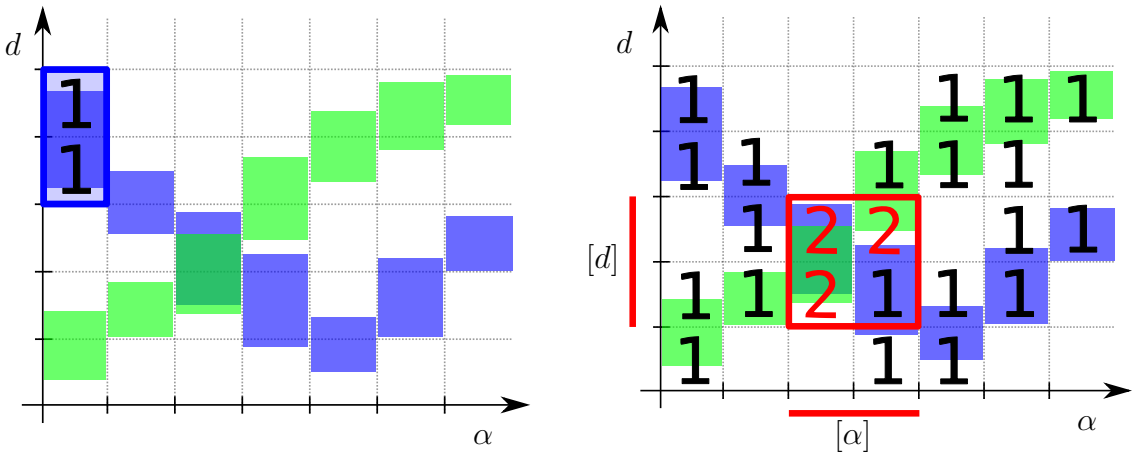
Figure 6.5: Interval-based Hough Transformation. From a set of boxes, an accumulator is filled using the angle-distance line parametrization. Each angle interval $[\alpha]$ generates a distance interval $[d]$ for a measurement $[\mathbf{m}]$.

we track the age of each particle in the coarse localization. Only if the considered particles have survived a minimal number of iterations it is considered as stable. As a result, we choose the particle with the highest weight that has survived a minimal number of iterations. This particle provides the position vector and the orientation value as scalar values. We can use the parameters to transform the local LiDAR point cloud to the map frame applying (6.3). Points that are close enough to facades are associated. Hence, we utilize a simple nearest-neighbor association representing a typical probabilistic method. Note that the association suppresses possible alternative solutions and only considers the most likely one. However, this method works well in practice since in the coarse localization particles form clusters in the feasible set very rapidly. In the scope of this work, we will only evaluate the probabilistic approach since, within the HyPaSCoRe Localization pipeline, this approach provides the best performance.

6.2.3 Interval-based Hough Transformation

The quality of the above-described association procedure severely depends on the uncertainty of the predicted pose estimate $[{}^{M}\varphi_{L_t}]$ and ${}^{M}\mathcal{T}_{L_t}$. The more uncertain the predicted pose estimate is, the larger the transformed boxes $[{}^{M}\mathbf{m}]$ will be in the case of the set-membership-based association. In the case of a probabilistic association, we may consider error-prone associations. That means more non-facade measurements are qualified as potential candidates for the facades. As a result, the associated clusters of local measurements, as illustrated in Figure 6.4c, will contain local measurements that are not part of the bare planes that describe the facades. This processing step aims to extract a line segment with the highest support from each associated box cluster. Furthermore, we need to consider the uncertainty of the extracted lines using intervals. Therefore, we introduce a novel interval-based Hough Transformation (iHT) similar to the approach presented by Jaulin in [145].

A line is parametrized by the angle and the orthogonal distance as described in (2.73). As our goal is to extract the line segment with the highest support from each associated box cluster, the Hough space, into which we need to transform the local measurements, is spanned by the two parameters. The Hough space is illustrated in the right part of Figure 6.5. In the



(a) Intervals in the accumulator. (b) Interval line extraction.

Figure 6.6: Interval line with the highest support from the accumulator. Figure 6.6a shows the approximation of an interval's upper and lower bound. Figure 6.6b shows the accumulator values. The interval line corresponds to the angle and distance intervals hull enclosing the highest accumulator values.

left part, an exemplary cluster of boxes is shown. All the boxes are associated with a facade according to the map selection procedure described above. Each clustered box is transformed to the Hough space by taking (2.73) into account. This transformation is illustrated in Figure 6.5 for two exemplary boxes colored green and blue. Note that each box in the measurement space generates a graph in the Hough space. However, to ensure an efficient representation, we approximate the Hough space using a quantized accumulator, shown in Figure 6.6a. The main advantage of the interval representation of the parameters is that intervals naturally comply with quantizations. Hence, the transformation of boxes from the measurement space to the Hough space is well represented by intervals. To build the graph in the Hough space for a given box $[\mathbf{m}] = \begin{pmatrix}[x] & [y]\end{pmatrix}$, we quantize the angle into smaller intervals. In our experiments, an interval width of $0.5°$ has proven to be a proper quantization. For each of the angle intervals $[\alpha]$ we determine the corresponding distance interval $[d]$ for the box $[\mathbf{m}]$ by taking the line constraint

$$\begin{pmatrix}\cos(\alpha) & \sin(\alpha)\end{pmatrix} \cdot \begin{pmatrix}x\\ y\end{pmatrix} - d = 0 \tag{6.5}$$

into account. By applying a forward-backward contractor, the interval $[d]$ is determined. We perform this operation for all angle intervals $[\alpha]$ and obtain the corresponding distance intervals $[d]$ given a measurement box $[\mathbf{m}]$. The resulting distance intervals to a corresponding angle interval are depicted in the right part of Figure 6.5. The green measurement box in the left part generates the sequence of green distance intervals, and the blue box generates the blue sequence.

A distance interval $[d]$ can partially overlap with a cell in the accumulator. The goal of the accumulator is to count the number of contributions of measurements to each cell, which represents a set of line parameters. To ensure an efficient accumulator representation, we overestimate the uncertainty by taking the lowest upper bound and the highest lower bound according to the quantization that includes the interval as shown in Figure 6.6a.

If we build the accumulator for all the measurements, the cells of the accumulator represent the support of a specific set of line parameters by the measured data. In our exemplary case, where we only transformed two measurement boxes to the Hough space, the resulting accumulator is shown in Figure 6.6b. In this illustration, we only show the non-zero entries in the accumulator. Note that the accumulator value is two for those cells where the green and blue graphs intersect. That means the blue and green measurement boxes support the corresponding sets of line parameters. Hence, as we seek to find the line parameters with the highest support in the measured data, we need to detect the cells with the highest accumulator values. Therefore, we take the accumulator cell with the highest support v (high accumulator cell value) and consider all the cells with an accumulator value higher than $0.95v$. The cells with high accumulator values will be close together for suitable line structures, as illustrated in Figure 6.6b. However, no significant peak will be detectable in the accumulator for bad line structures where the box cluster does not form a line-like structure. That is why we only select clusters of cells in an accumulator supported by enough measurements (minimal cell threshold) and have a significant peak so that $0.95v$ is five times higher than the average non-zero cell value. The parameters are chosen empirically. This restrictive selection procedure only considers cells with high support, and the extraction of the line parameters becomes robust against artifacts in the measurements. In Figure 6.6b, the cell values of selected cells are colored red. In this simple example, all the cells with an accumulator value of two are further considered. We then take the hull of all selected cells in the Hough space so that we get an interval for the angle $[\alpha]$ and for the distance $[d]$ as illustrated in Figure 6.6b by a red box. These parameters describe the set of lines with the highest support and incorporate the uncertainty of contributing measurements.

Computing the accumulator of the hough space can become heavy in computation. However, we can reduce the size of the Hough space by considering prior knowledge. By transforming the associated map line from the map frame to the local frame using the initial pose estimate, we obtain comparatively uncertain estimates for the local line parameters. However, the first estimate of the local line parameter reduces the Hough space as we exploit the information where we expect a line to be in the local data. This reduces computational costs as the accumulator can be set smaller. On the downside, we are implicitly deciding which facade a box cluster has to belong to. Consequently, ambiguous localization is suppressed. However, this approach works well in practice with real data if the initial pose uncertainty is small enough to avoid ambiguous matching.

As a result, for each box cluster the iHT determines the set of lines defined by the angle $[^L\alpha]$ and the distance $[^Ld]$ that have the highest support. Furthermore, we can determine the subset of boxes consistent with this parameter set using the accumulator as we track which of the boxes contribute to which cell. Consequently, by choosing the two boxes $[^L\mathbf{a}_1]$ and $[^L\mathbf{a}_2]$ among the supporting boxes that have the highest distance to each other we can describe the line as a line segment – similar to the facade F_i. As a result, the iHT provides a local line segment $L_i = \{[^L\mathbf{a}_1], [^L\mathbf{a}_2], [^L\alpha], [^Ld]\}$ that is associated with the facade F_i.

6.2.4 Orientation Contraction

For each associated pair L_i and F_i, the orientation is determined by the difference between the line angles $[{}^M\alpha_i]$ and $[{}^L\alpha_i]$. For n associated pairs, the orientation interval is determined by

$$[{}^M\varphi_{L_t}] = [{}^M\varphi_{L_t}] \bigcap_{i\in\{1,\dots,n\}} [{}^M\alpha_i] - [{}^L\alpha_i]. \tag{6.6}$$

Note that two equivalent parameter sets can represent a line expressed by the angle and distance parametrization since a line can be rotated by π. However, since we restrict the Hough space based on the expected parameter set by transforming the associated map line to the local frame, we ensure that the angle representation complies with the angle of the map line. By doing this, we also bypass the modulo-2π problem.

6.2.5 Position Polygon

The distance parameter of the extracted line L_i provides the orthogonal distance information to the map line F_i. This means the vehicle's position is restricted along the normal direction of the facade F_i. In other words, the associated pair L_i and F_i provides a stripe that is oriented in parallel to F_i, which restricts the position in the normal direction of F_i. We illustrate this constraint in Figure 6.7 by a green stripe. While the black line segment represents F_i and the blue line segment L_i, the green arrow illustrates the constraint in the lateral direction that generates the green stripe that restricts the vehicle position. However, the normal direction of F_i is not fixed since the angle parameter $[{}^M\alpha_i]$ is an interval. Nonetheless, this parameter mainly depends on the quality of the map. In practice, the interval width of $[{}^M\alpha_i]$ is typically small since the orientation uncertainty is small for the building maps that we use. Consequently, we can approximate the orientation interval $[{}^M\alpha_i]$ of F_i by a scalar orientation by taking the mid ${}^M\alpha_{\text{mid}}$. As a result, F_i is approximated by a set of parallel lines as the orientation is approximated to a fixed scalar. Accordingly, the position $\begin{pmatrix} t_x & t_y \end{pmatrix}^T$ has to satisfy

$$\begin{pmatrix} \cos({}^M\alpha_{\text{mid}}) & \sin({}^M\alpha_{\text{mid}}) \end{pmatrix} \cdot \begin{pmatrix} t_x \\ t_y \end{pmatrix} = {}^Md - {}^Ld = d_\perp, \tag{6.7}$$

for unknown $t_x \in [t_x]$ and $t_y \in [t_y]$ and known ${}^Md \in [{}^Md]$, ${}^Ld \in [{}^Ld]$ and $d_\perp \in [d_\perp]$. That means an upper and lower line parallel to the facade can border the position. The position has to be within the stripe as illustrated with the green stripe in Figure 6.7.

We can draw a further constraint from the association of L_i and F_i. The finite length of the line segments constrains the position along the facade F_i because L_i and F_i need to intersect. The orange arrow in Figure 6.7 illustrates this constraint. This means the line intersection constraint restricts the set of feasible positions in the perpendicular direction of F_i. The feasible region based on this constraint is colored orange. An interval line equation can also define the orange stripe. Since the stripe is oriented perpendicular to F_i, the orientation of the stripe is also approximated by a scalar angle determined by ${}^M\alpha_{\text{mid}} + \frac{\pi}{2}$. However, the distance parameter, which depends on the start and end points of the line segments, needs to be determined. Therefore, we calculate the parameter $[{}^gd_\parallel]$ that describes the set of distances

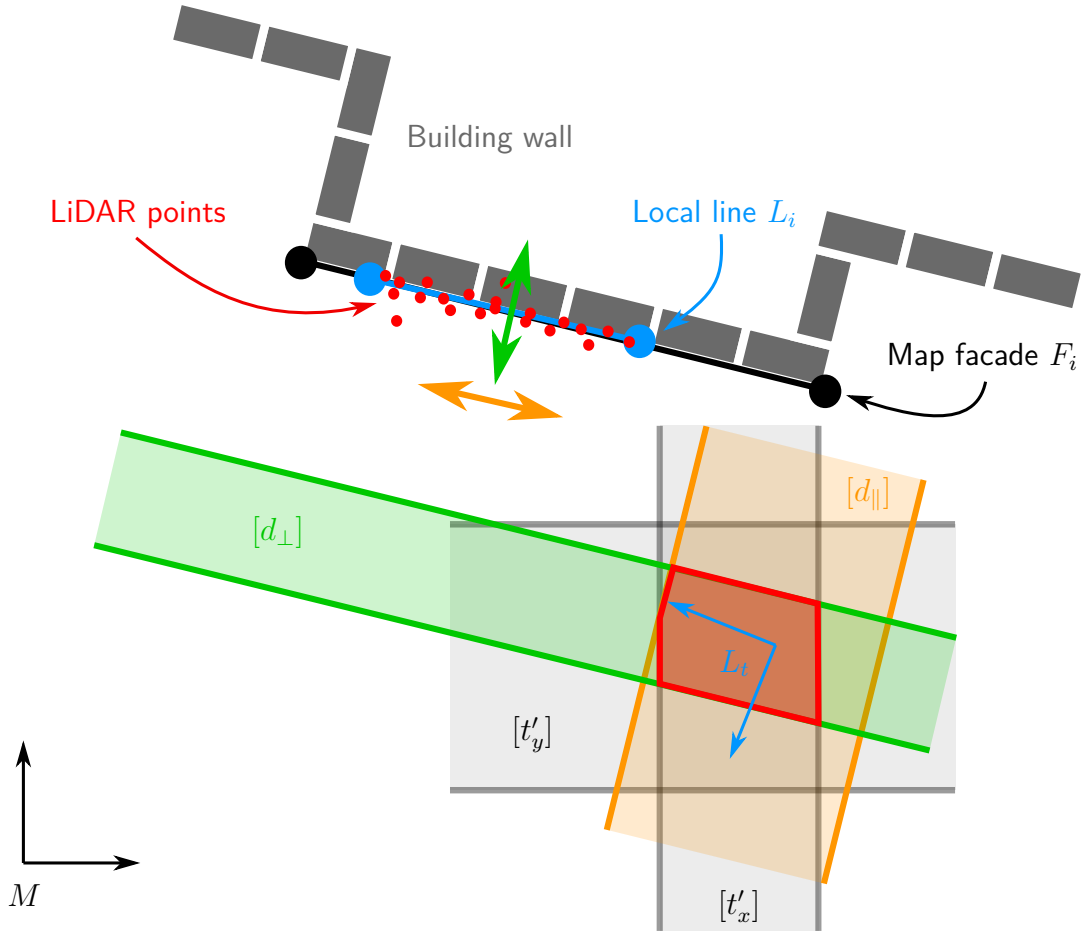


Figure 6.7: Matched line segments and the predicted position provide constraints. The map facade F_i is colored black, and the local line segment L_i is blue. The position is constrained along the normal of the line segment (green) and along the line due to finite segment lengths (orange). Each line segment match provides two parallel lines that border the feasible position. The predicted vehicle position further constrains the consistent set visualized by gray stripes. The red polygon describes the region consistent with the facade association and the predicted position.

of two arbitrary points on the line between the start point ${}^g\mathbf{a}_1$ and endpoint ${}^g\mathbf{a}_2$ for the map line segment and a local line segment ($g \in \{m, l\}$). As we consider interval uncertainty, the distance interval is defined by the hull

$$[{}^g d_\parallel] = [{}^g\mathbf{n}^T] \cdot [{}^g\mathbf{a}_1] \sqcup [{}^g\mathbf{n}^T] \cdot [{}^g\mathbf{a}_2], \tag{6.8}$$

for ${}^g\mathbf{n} = \begin{pmatrix}\cos({}^g\alpha + \frac{\pi}{2}) & \sin({}^g\alpha + \frac{\pi}{2})\end{pmatrix}^T$, ${}^g\alpha \in [{}^g\alpha]$ and ${}^g\mathbf{n} \in [{}^g\mathbf{n}]$ based on the corner points of the line segments L_i and F_i. Now we can describe the orange stripe by

$$\begin{pmatrix}\cos({}^M\alpha_{\text{mid}} + \frac{\pi}{2}) & \sin({}^M\alpha_{\text{mid}} + \frac{\pi}{2})\end{pmatrix} \cdot \begin{pmatrix}t_x \\ t_y\end{pmatrix} = d_\parallel, \tag{6.9}$$

for known $d_\parallel = {}^M d_\parallel - {}^L d_\parallel$, ${}^M d_\parallel \in [{}^M d_\parallel]$ and ${}^L d_\parallel \in [{}^L d_\parallel]$ and unknown $t_x \in [t_x]$ and $t_y \in [t_y]$. This constraint again introduces a stripe represented by an upper and lower line perpendicular

to the facade. As a result, each associated facade gives a stripe (pair of bordering lines) parallel and perpendicular to the facade as shown in Figure 6.7.

Nevertheless, not only the associated facades constrain the set of positions. Also, the predicted set of positions ${}^{M}\mathcal{T}_{L_t}$ deduced from the previous estimate ${}^{M}\mathcal{T}_{L_{t-1}}$ using odometry data constrains the set of positions. As introduced in Subsection 6.2.1, $\begin{pmatrix}[t'_x] & [t'_y]\end{pmatrix}^T = {}^{M}\mathcal{T}_{L_t}$ is represented by an interval box. As the predicted box encloses the consistent set of positions, $[t_x] \subset [t'_x]$ and $[t_y] \subset [t'_y]$ hold as further constraints on the set of positions. Since ${}^{M}\mathcal{T}_{L_t}$ is axis-aligned, we can represent the constraints in the same stripe form as for the facade association as illustrated in Figure 6.7. Hence, the predicted position in the x-direction generates the stripe constraint

$$\begin{pmatrix}1 & 0\end{pmatrix} \cdot \begin{pmatrix}t_x \\ t_y\end{pmatrix} = t'_x, \tag{6.10}$$

with $t'_x \in [t'_x]$. The predicted position in the y-direction can be formulated as the constraint

$$\begin{pmatrix}0 & 1\end{pmatrix} \cdot \begin{pmatrix}t_x \\ t_y\end{pmatrix} = t'_y, \tag{6.11}$$

with $t'_y \in [t'_y]$. Both are visualized in Figure 6.7 by gray-colored stripes.

The intersection region of all the stripes represents the region of positions consistent with the facade association and the predicted position. This region can be described by a convex polygon and is illustrated in Figure 6.7 in red. That means since each seen facade in the environment provides two stripes, in the case of m seen facades, we obtain $2m + 2$ stripes. The $+2$ stems from the predicted position using odometry data. The intersection of all $2m + 2$ stripes defines the feasible set of positions taking the observations and their uncertainties into account.

We can formulate (6.7) and (6.9) for multiple facades as an interval linear equation system

$$\begin{pmatrix}\cos(\alpha_1) & \sin(\alpha_1) \\ \vdots & \vdots \\ \cos(\alpha_n) & \sin(\alpha_n)\end{pmatrix} \begin{pmatrix}t_x \\ t_y\end{pmatrix} = \begin{pmatrix}d_1 \\ \vdots \\ d_n\end{pmatrix}, \tag{6.12}$$

where each facade contributes two rows, $t_x \in [t_x]$ and $t_y \in [t_y]$ are unknowns, and $d_1 \in [d_1]$ and $d_n \in [d_n]$ are described by known intervals. Note that due to the approximation of the orientation uncertainty of the map facade by a scalar, the matrix on the left side of the equation is a matrix with scalar entries. Hence we can rewrite (6.12) to the common form $\mathbf{A} \cdot \mathbf{t} = \mathbf{d}$ with $\mathbf{t} \in [\mathbf{t}]$ and $\mathbf{d} \in [\mathbf{d}]$. In interval analysis, efficient tools exist to determine the smallest axis-aligned box $\begin{pmatrix}[t_x] & [t_y]\end{pmatrix}^T$ enclosing the solution space that satisfies the linear equation system (6.12) [49]. However, we want to determine the exact, minimal polygon described by the corner points. Therefore, we propose a novel method illustrated in Figure 6.8 exploiting the parallel border lines of each stripe. The corresponding algorithm is presented in Algorithm 8.

Each row in (6.12) corresponds to a stripe bordered by an upper and lower line defined by the interval distance $[d_i]$. As a consequence, from each row, we obtain two parallel lines. In Figure 6.8, we illustrate three stripes (red, blue, and green) with the upper and lower border lines. We first determine all possible intersections between the bordering lines as shown in

Algorithm 8: Polygon Corner Extraction

Data: Linear interval equation system $\mathbf{A} \cdot \mathbf{t} = \mathbf{d}$ with $\mathbf{t} \in [\mathbf{t}]$ and $\mathbf{d} \in [\mathbf{d}]$ as presented in (6.12)

Result: Set of corner points $\mathcal{C}$, that defines a polygon enclosing consistent set of positions $[\mathbf{t}]$

Function *LineIntersection*($\mathbf{n}_1 = \begin{pmatrix} n_{1,x} & n_{1,y} \end{pmatrix}^T$,$d_1$,$\mathbf{n}_2 = \begin{pmatrix} n_{2,x} & n_{2,y} \end{pmatrix}^T$,$d_2$)

  `// Compute the intersection of two lines described in the normal form.`

  $y = \frac{d_2 - \frac{n_{2,x}}{n_{1,x}} \cdot d_1}{n_{2,y} - \frac{n_{2,x}}{n_{1,x}} \cdot n_{1,y}}$;

  $x = \frac{d_1 - n_{1,y} \cdot y}{n_{1,x}}$;

  return $\begin{pmatrix} x & y \end{pmatrix}^T$;

Function *FeasiblePositionPolygon*($\mathbf{A}$,$[\mathbf{d}]$)

  `// Initialize empty set of corner points`

  $\mathcal{C} = \varnothing$;

  `// Compute all intersections between the border lines of the stripes`

  for $i = 0$; $i <$ *len*$([\mathbf{d}])$; $i++$ **do**

    for $j = i + 1$; $j <$ *len*$([\mathbf{d}])$; $j++$ **do**

      `// Get first stripe`

      $\mathbf{n}_i = \mathbf{A}$.getColumn$(i)^T$;

      $[d_i] = [\mathbf{d}]$.getColumn(i);

      `// Get second stripe`

      $\mathbf{n}_j = \mathbf{A}$.getColumn$(j)^T$;

      $[d_j] = [\mathbf{d}]$.getColumn(j);

      `// Compute intersections`

      $\mathbf{p}_1 =$ LineIntersection$(\mathbf{n}_i, [d_i]$.lb(), $\mathbf{n}_j, [d_j]$.lb()$)$;

      $\mathbf{p}_2 =$ LineIntersection$(\mathbf{n}_i, [d_i]$.lb(), $\mathbf{n}_j, [d_j]$.ub()$)$;

      $\mathbf{p}_3 =$ LineIntersection$(\mathbf{n}_i, [d_i]$.ub(), $\mathbf{n}_j, [d_j]$.lb()$)$;

      $\mathbf{p}_4 =$ LineIntersection$(\mathbf{n}_i, [d_i]$.ub(), $\mathbf{n}_j, [d_j]$.ub()$)$;

      $\mathcal{C}$.push_back$(\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\})$;

    end

  end

  `// Check which of the intersection points are inside the stripes`

  for $i = 0$; $i <$ *len*$([\mathbf{d}])$; $i++$ **do**

    $\mathbf{n}_i = \mathbf{A}$.getColumn$(i)^T$;

    $[d_i] = [\mathbf{d}]$.getColumn(i);

    $\alpha_i =$ atan2$(n_{i,y}, n_{i,x})$;

    $\mathbf{R} = \begin{pmatrix} \cos(\alpha_i) & -\sin(\alpha_i) \\ \sin(\alpha_i) & \cos(\alpha_i) \end{pmatrix}$;

    for $j = 0$; $j <$ *len*$(\mathcal{C})$; $j++$ **do**

      `// Rotate the intersection point to stripe-oriented frame`

      $\hat{\mathbf{p}} = \mathbf{R} \cdot \mathcal{C}[j]$;

      if *not*($[d_i]$.*contains*$(\hat{p}_x)$) **then**

        `// p is not inside the stripe, delete p from` $\mathcal{C}$

        $\mathcal{C}$.erase(j);

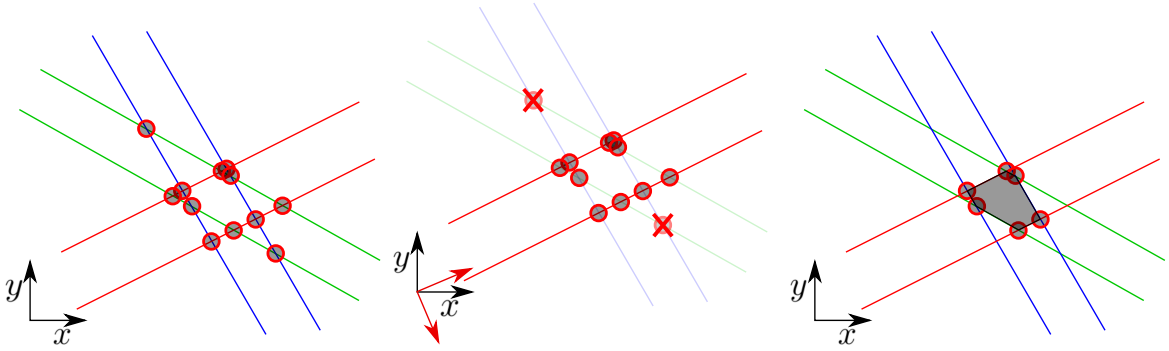
        $j--$;

      end

    end

  end

  return $\mathcal{C}$



(a) Intersections. (b) Deletion. (c) Polygon.

Figure 6.8: Determine the polygon from a set of stripes. Figure 6.8a shows all intersections of the bordering lines of the stripes. Iteratively deleting the points if they do not lie within the stripes as illustrated in Figure 6.8b, the polygon corners are determined as shown in Figure 6.8c, that define the borders of the feasible solution set.

Figure 6.8a with red circles. In Algorithm 8, the computation of all intersection points is shown in lines 7 to 19. Note that the line intersection function is defined in lines 1 to 4. Each of the intersection points is inserted into a set of potential corner points $\mathcal{C}$ (line 17).
The next goal is to retrieve the intersection points that are part of the common intersection region of all stripes since these corner points define the polygon corners. Therefore, for each of the stripes we check if an intersection point lies within the stripe. If it is not located inside the region, the corner will be deleted from the set of corner points $\mathcal{C}$. We illustrate this in Figure 6.8b for the red stripe. The deletion process is described in lines 20 to 32 in Algorithm 8. To check if a point is located inside the stripe, we first transform the point to a rotated coordinate frame located in the origin of the original frame. The rotated coordinate frame is oriented in such a way that the x-axis points to the orthogonal (normal) direction of the considered stripe. In Figure 6.8b, this rotated coordinate frame is colored red. The corresponding rotation matrix is fully determined by the angle α_i as shown in lines 23 and 24 in Algorithm 8. The rotation is applied to a test point in line 26. The transformation simplifies the test if a point is inside or outside the stripe to an efficient scalar in interval test: As the x-component of the transformed point $\hat{\mathbf{p}}$ defines the distance of the point to the frame origin along the stripe normal, the x-component has to be inside the distance interval $[d_i]$ if the point is inside the stripe. If this does not hold, the considered point will be erased from the set of corner points $\mathcal{C}$ (line 28). This procedure is performed for each row of the system (6.12) – ergo for each stripe. The remaining set $\mathcal{C}$ defines the corner points of the polygon as shown in Figure 6.8c.

Conclusively, we want to provide further insights into the runtime complexity of this algorithm. For n different stripes, $2n$ border lines potentially intersect. However, each pair of lines of a stripe cannot intersect with its partner line and with itself. Hence, for the number of intersection points we obtain the following mathematical series

$$2 \cdot (2n-2) + 2 \cdot (2n-4) + \ldots + 2 \cdot (2n - 2 \cdot (n-1)) = \sum_{m=1}^{n-1} 4 \cdot (n-m). \tag{6.13}$$

Note that the first term on the left side of the equation provides the number of intersection points when considering the first stripe: Let us consider one line of the stripe. In total, $2n$ intersections are possible, but as the line cannot intersect with its partner in the stripe (parallel) and cannot intersect with itself, two must be subtracted. As this number of intersections also applies to the partner line in the stripe, we obtain $2 \cdot (2n-2)$. We can argue similarly for the next stripe, but we need to subtract two further intersections as those are already considered in the first term. Hence, we obtain the term $2 \cdot (2n-4)$. We obtain the series mentioned above by summing up the terms for all n stripes. We can rewrite (6.13) to

$$\sum_{m=1}^{n-1} 4 \cdot (n-m) = 4 \cdot \Big(\sum_{m=1}^{n-1} n - \sum_{m=1}^{n-1} m \Big) = 4 \cdot \Big(n \cdot (n-1) - \frac{n \cdot (n-1)}{2} \Big) = 2 \cdot (n^2 - n) \quad (6.14)$$

applying the Gaussian summation formula. As a result, the number of intersections rises quadratically with the number of stripes. Since we need to perform at maximum an intersection-in-stripe check per stripe, we can bound the number of checks by $2 \cdot (n^3 - n^2)$. That means the runtime complexity of our algorithm to solve the linear interval equation system is $\mathcal{O}(n^3)$. Fortunately, in our application, the number of facades observed in one frame is limited. Accordingly, only a few intersection points need to be considered, making our approach well applicable to the localization problem in building maps. The computed polygon describes the minimal set concerning the linear equation system (6.12).

6.3 Maximum Likelihood Estimation with Rigid Bounds

The interval-based method provides a set of poses consistent with local observations and the building map. The consistent set is defined by an interval $[{}^M\varphi_L]$ for the orientation and a polygon ${}^M\mathcal{T}_L$ for the set of positions. A set of corners describes the polygon points $\mathcal{C}$. This section aims to determine the most likely solution within the consistent set. Therefore, some previously made assumptions must be adapted: We initially assumed that all errors are bounded to apply the interval-based approach. As a result, in the sense of set-membership-based approaches, there is no possibility of determining the most viable solution since all poses in the consistent set are equally likely. However, we cannot draw any conclusions on the likelihood of individual solutions. That is why we need to refine our error model further. In the scope of this work, we augment our assumption on the boundedness of measurement errors by a Gaussian distribution of the error. This provides us with two advantages. First, the boundedness assures that we can restrict the search space to the consistent set of poses provided by the interval-based refinement approach. Second, an MLE under normally distributed errors leads to a least-squares formulation of the problem that can be solved by well-known methods like Newton or Levenberg-Marquardt (LM) [10]. In summary, this section aims to formulate the localization problem as a least-squares problem and to solve the minimization problem with the LM method under consideration of the rigid bounds defined by the interval-based refinement method.

First, we must define the objective function we seek to minimize. Therefore, let us again consider our measurements and observations: We measure points in our environment with a 3D LiDAR sensor. Further, we have a building map of our environment that provides us with

information on the location of the facade in the map frame. However, more information is available: In the interval-based refinement, we performed an association of the local LiDAR measurements to the facades (Subsection 6.2.2) and then we select the most supportive local measurements for each facade utilizing the iHT (Subsection 6.2.3). Consequently, the interval-based refinement provides us – as a side-product – the information on which local measurement is associated with which facade in the map. Let ${}^{L}\mathbf{m}$ be a local measurement in the LiDAR frame that is associated with facade F_i which is described by the normal vector ${}^{M}\mathbf{n}$ and the distance parameter ${}^{M}d$. As a result, we seek to find a pose ${}^{M}\mathbf{T}_{L}$ so that

$$ {}^{M}\mathbf{n}^{T} \cdot ({}^{M}\mathbf{T}_{L} \cdot {}^{L}\mathbf{m}) = {}^{M}d \tag{6.15} $$

holds. Now let us write out the vectors' and matrix's individual elements in (6.15).

$$ \begin{pmatrix} n_x & n_y \end{pmatrix} \cdot \left(\begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix} \cdot \begin{pmatrix} m_x \\ m_y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \right) = {}^{M}d. \tag{6.16} $$

By further expanding the equation, we obtain

$$ \cos(\varphi) \cdot (m_x \cdot n_x + m_y \cdot n_y) + \sin(\varphi) \cdot (m_x \cdot n_y - m_y \cdot n_x) + t_x \cdot n_x + t_y \cdot n_y = {}^{M}d. \tag{6.17} $$

While n_x, n_y, ${}^{M}d$, m_x and m_y are known from the map and the local LiDAR measurement, the pose parameters $\mathbf{p} = \begin{pmatrix} t_x & t_y & \varphi \end{pmatrix}^{T}$ are the unknowns. We seek to find a parameter set $\mathbf{p}^{*}$ that fulfills (6.17) as well as possible. To cast the problem to a least-squares problem, we define the left part of (6.17) as the *virtual measurement* function $f(\mathbf{p})$. Hence, the *real measurement* x is defined by the right term ${}^{M}d$ of (6.17). That means we define our measurement based on the robot's distance to a seen facade along the normal direction. However, (6.17) only considers one LiDAR point associated with one facade. In each frame, multiple points are associated with one facade, while we also observe multiple facades. As a consequence, we need to consider multiple constraints of the same form as presented in (6.17) and the multidimensional *virtual measurement* function $\mathbf{f}(\mathbf{p})$ and the measurement vector $\mathbf{x}$ are defined by

$$ \mathbf{f}(\mathbf{p}) = \begin{pmatrix} f_{1,1}(\mathbf{p}) \\ f_{1,2}(\mathbf{p}) \\ \vdots \\ f_{1,p_1}(\mathbf{p}) \\ f_{2,1}(\mathbf{p}) \\ \vdots \\ f_{2,p_2}(\mathbf{p}) \\ \vdots \\ f_{n,1}(\mathbf{p}) \\ \vdots \\ f_{n,p_n}(\mathbf{p}) \end{pmatrix} \text{ and } \mathbf{x} = \begin{pmatrix} {}^{M}d_1 \\ {}^{M}d_1 \\ \vdots \\ {}^{M}d_1 \\ {}^{M}d_2 \\ \vdots \\ {}^{M}d_2 \\ \vdots \\ {}^{M}d_n \\ \vdots \\ {}^{M}d_n \end{pmatrix}, \tag{6.18} $$

where we stack all virtual measurement functions and real measurements. The lower right index of $f_{n,p_n}(\mathbf{p})$ identifies the virtual measurement function of the n-th seen facade and the p_n-th point associated with that facade. Now we can define the minimization problem as

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} ||\mathbf{x} - \mathbf{f}(\mathbf{p})||^2. \tag{6.19}$$

We apply a modified version of the LM non-linear least-squares algorithm to solve this problem. Therefore, we need to determine the Jacobian of $\mathbf{f}(\mathbf{p})$ concerning $\mathbf{p}$. We define $\mathbf{J} = \frac{\delta \mathbf{f}(\mathbf{p})}{\delta \mathbf{p}}$. For the i-th facade and the j-th point on that facade, the Jacobian matrix $\mathbf{J}_{ij}$ can be derived from (6.17). We omit the indices i and j for simplicity. As a result, we obtain

$$\mathbf{J}_{ij} = \begin{pmatrix} \frac{\delta f_{i,j}(\mathbf{p})}{\delta t_x} & \frac{\delta f_{i,j}(\mathbf{p})}{\delta t_y} & \frac{\delta f_{i,j}(\mathbf{p})}{\delta \varphi} \end{pmatrix} \tag{6.20}$$

$$= \begin{pmatrix} n_x & n_y & -\sin(\varphi)(m_x \cdot n_x + m_y \cdot n_y) + \cos(\varphi)(m_x \cdot n_y - m_y \cdot n_x) \end{pmatrix}. \tag{6.21}$$

That means each measurement contributes one row to $\mathbf{J}$.

The modified LM algorithm is presented in Algorithm 9. The original LM algorithm was adopted from [23]. However, the original version poses two problems to our application. First, it does not consider the rigid bounds of the feasible set of pose parameter $\mathcal{P}$ when the update of the parameter vector is computed. Second, one degree of freedom is not sufficiently constrained as the point-to-plane constraint in (6.17) only considers the error in the orthogonal direction while along the facade, the pose is not constrained. Due to this problem, the optimizer may shift the poses along the facade direction, as the update vector may have non-zero parts in this direction. Therefore we introduce modifications to the LM algorithm that will be discussed in the following in detail.

As input data, we provide the optimizer with the virtual measurement function $\mathbf{f}(\mathbf{p})$, the measurement vector $\mathbf{x}$, the set of normal vectors $\mathcal{N}$ of the seen facades, the feasible set of poses $\mathcal{P} = \{{}^M\mathcal{T}_L, [{}^M\varphi_L]\}$ computed by the interval-based refinement and the initial pose parameters $\mathbf{p}_0$. We choose $\mathbf{p}_0$ as the mid of $\mathcal{P}$. We define the mid of a polygon ${}^M\mathcal{T}_L$ as the midpoint of the box-hull, while for the orientation, the interval's midpoint is chosen. The key ingredient of the damped gradient descent approach is the Jacobian matrix $\mathbf{J}$ that we already introduced for our problem above. Note that the Jacobian is always computed after updating the pose parameter $\mathbf{p}$ (lines 3 and 22). That means the linearization is reevaluated for each iteration step, which qualifies the LM algorithm to deal with non-linearities. The system matrix $\mathbf{A}$ and the error $\boldsymbol{\epsilon}_\mathbf{p}$ are also computed for each update of $\mathbf{p}$ as shown in lines 4 and 23. For each iteration step the update vector $\hat{\boldsymbol{\delta}}_\mathbf{p}$ is computed in line 9 by solving the linear system. Note that the LM algorithm solves the augmented normal equations where a damping term μ controls the size of the update vector $\hat{\boldsymbol{\delta}}_\mathbf{p}$. In line 5, μ is initialized with the maximal diagonal value of the system matrix $\mathbf{A}$ times a factor τ. However, solving the system in line 9 will lead to an update vector $\hat{\boldsymbol{\delta}}_\mathbf{p}$ that may contain parts along the non-constrained directions due to the unconstrained degree of freedom for the point-to-line matching as stated above. Consequently, the translation part of $\hat{\boldsymbol{\delta}}_\mathbf{p}$ – the first two parameters – needs to be corrected. The correction is introduced in Algorithm 9 in lines 10 to 14. While the orientation update remains unchanged (line 10), the translation part is corrected based on the normal vectors of all seen facades. As we only should consider the update directions which point along the normal vectors of the

Algorithm 9: Levenberg-Marquardt Optimization with rigid bounds

Data: Virtual measurement function $\mathbf{f}(\mathbf{p})$, measurement vector $\mathbf{x}$, set of normal vectors $\mathcal{N} = \{\mathbf{n}_{m,1}, \ldots, \mathbf{n}_{m,n}\}$ for each seen facade, the consistent set of poses $\mathcal{P} = \{{}^{M}\mathcal{T}_L, [{}^{M}\varphi_L]\}$ initial parameters $\mathbf{p}_0$.
Result: The vector $\mathbf{p}^*$ minimizing $||\mathbf{x} - \mathbf{f}(\mathbf{p})||^2$.

Function *LevenbergMarquardtOptimizationWithRigidBounds*($\mathbf{f}(\mathbf{p})$, $\mathbf{x}$, $\mathcal{N}$, $\mathcal{P}$, $\mathbf{p}_0$)
  $k = 0$; $v = 2$; $\mathbf{p} = \mathbf{p}_0$;
  $\mathbf{J} =$ computeJacobian$(\mathbf{p})$;
  $\mathbf{A} = \mathbf{J}^T\mathbf{J}$; $\boldsymbol{\epsilon}_\mathbf{p} = \mathbf{x} - \mathbf{f}(\mathbf{p})$; $\mathbf{g} = \mathbf{J}^T\boldsymbol{\epsilon}_\mathbf{p}$;
  stop $= (||\mathbf{g}||_\infty \leq \epsilon_1)$; $\mu = \tau \cdot \max(A_{ii})$;
  while *(not stop) and* $(k < k_{max})$ **do**
    $k = k + 1$;
    do
      `// Compute the update`
      Solve $(\mathbf{A} + \mu\mathbf{I})\hat{\boldsymbol{\delta}}_\mathbf{p} = \mathbf{g}$;
      `// Correct the translation part of the update`
      $\boldsymbol{\delta}_\mathbf{p} = \begin{pmatrix} 0 & 0 & \hat{\boldsymbol{\delta}}_\mathbf{p}(2) \end{pmatrix}^T$;
      for $\mathbf{n}_{m,i} : \mathcal{N}$ **do**
        $\hat{\boldsymbol{\delta}}_\mathbf{t} = \hat{\boldsymbol{\delta}}_\mathbf{p}$.subvector$(0, 1)$;
        `// Only consider partial update along the normal vector` $\mathbf{n}_{m,i}$
        $\boldsymbol{\delta}_\mathbf{p}$.subvector$(0, 1) = \boldsymbol{\delta}_\mathbf{p}$.subvector$(0, 1) + \frac{\mathbf{n}_{m,i}^T \cdot \hat{\boldsymbol{\delta}}_\mathbf{t}}{||\mathbf{n}_{m,i}||} \cdot \mathbf{n}_{m,i}$;
      end
      if $||\boldsymbol{\delta}_\mathbf{p}|| \leq \epsilon_2 ||\mathbf{p}||$ **then**
        `// Update is not significant enough, so stop`
        stop $=$ true;
      else
        $\mathbf{p}_{\text{new}} = \mathbf{p} + \boldsymbol{\delta}_\mathbf{p}$;
        $\rho = \frac{||\boldsymbol{\epsilon}_\mathbf{p}||^2 - ||\mathbf{x} - \mathbf{f}(\mathbf{p}_{\text{new}})||^2}{\boldsymbol{\delta}_\mathbf{p}^T(\mu\boldsymbol{\delta}_\mathbf{p} + \mathbf{g})}$;
        if $\rho > 0$ *and* $\mathbf{p}_{new} \in \mathcal{P}$ **then**
          `// Error smaller and` $\mathbf{p}_{\text{new}}$ `is inside` $\mathcal{P}$`, take update`
          $\mathbf{p} = \mathbf{p}_{\text{new}}$;
          $\mathbf{J} =$ computeJacobian$(\mathbf{p})$;
          $\mathbf{A} = \mathbf{J}^T\mathbf{J}$; $\boldsymbol{\epsilon}_\mathbf{p} = \mathbf{x} - \mathbf{f}(\mathbf{p})$; $\mathbf{g} = \mathbf{J}^T\boldsymbol{\epsilon}_\mathbf{p}$;
          stop $= (||\mathbf{g}||_\infty \leq \epsilon_1)$;
          $\mu = \mu \cdot \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$; $v = 2$;
        else
          `// Error larger or` $\mathbf{p}_{\text{new}}$ `is not inside` $\mathcal{P}$
          `// Do not take update and increase damping factor`
          $\mu = \mu \cdot v$; $v = 2v$;
        end
      end
    while $(\rho > 0)$ *or stop*;
  end
  $\mathbf{p}^* = \mathbf{p}$;

seen facades, we can calculate the correct translation update vector as the sum over the scalar products of the translation update vector $\hat{\boldsymbol{\delta}}_{\mathbf{t}}$ and the normal vectors $\mathbf{n}_{m,i}$ as stated in line 13. As a result, at the end of the for-loop, we obtain a corrected update vector $\boldsymbol{\delta}_{\mathbf{p}}$.
First, we check in line 15 if the update vector is significantly large. Suppose the norm of the update vector is smaller than the adaptive threshold, we can infer that the gradient is close to zero. Therefore, we cannot significantly improve the current parameter estimate $\mathbf{p}$ and the algorithm can terminate. However, if the update is significant enough, we continue with line 18. Here, we compute the updated pose parameter $\mathbf{p}_{\text{new}}$ to check in line 19 if this updated parameter vector leads to a better solution. Note that in line 19 the variable ρ will be positive in the case that $\mathbf{p}_{\text{new}}$ reduces the error $||\mathbf{x} - \mathbf{f}(\mathbf{p}_{\text{new}})||^2$ compared to the old parameter vector. But on the other hand, if ρ is negative, the old error $||\boldsymbol{\epsilon}_{\mathbf{p}}||^2$ generated by the old parameter vector $\mathbf{p}$ is smaller than the error produced by $\mathbf{p}_{\text{new}}$. The LM algorithm takes the update only if $\mathbf{p}_{\text{new}}$ indeed reduces the error – that means if $\rho > 0$. If this is not the case, the damping factor μ increases. However, for our parameter estimation approach, we have a further criterion that has to be met: The updated parameter vector $\mathbf{p}_{\text{new}}$ needs to lie inside the feasible set $\mathcal{P}$. That is why we further modify the LM algorithm in line 20, where we only permit to update $\mathbf{p}$ if the $\rho < 0$ *and* if $\mathbf{p}_{\text{new}} \in \mathcal{P}$. We check if $\mathcal{P}$ contains $\mathbf{p}_{\text{new}}$ by first determining if the orientation parameter $\mathbf{p}_{\text{new}}(2)$ is inside $[{}^{M}\varphi_{L}]$ and second, performing the winding-number check to ensure that the translation part $\mathbf{p}_{\text{new}}$.subvector$(0,1)$ is inside ${}^{M}\mathcal{T}_{L}$.

The main advantage of the modified LM optimization with rigid bounds is that we can perform a joint optimization of the pose in one optimization procedure taking all facades into account. Nonetheless, the update vector correction in lines 10 to 14 may lead to divergence of the optimization if there is an unfortunate distribution of the points among the facades: Since we take individual point measurements into account, the number of points determines the weight of the different facades in the optimization. Hence, a facade with fewer points will have less impact on the update vector. If the facades are differently oriented, we also allow the update vector to include components in the normal direction of the facades with fewer points. The facades with more points will also incorrectly contribute to the update in this direction. Fortunately, in practice this happens rarely, but the optimization can diverge in those cases. However, the bounds can protect the optimization from significantly diverging, as the parameter vector cannot exit the feasible region. Consequently, the interval method provides an upper bound on the error. That means with the optimization with rigid bounds, we can guarantee – if and only if all assumptions are fulfilled – that the bounds provide the maximal error we can obtain from the localization.

6.4 Module Output

The first module in our localization pipeline is the visual odometry which computes the relative motion of the vehicle. As we do not know the vehicle's initial location in the building map, we perform the coarse localization in the second module. The coarse localization provides a feasible set of poses where the vehicle can be located, considering simple but always valid constraints. By augmenting the set-membership-based approach with a bounded MCL, the module provides the most likely locations of the vehicle in the building map. However, the

localization estimates are pessimistic for the set-membership result that encloses the feasible set of poses. As the set-membership result does not consider the direct associations of the LiDAR data to the building map, the full potential of the sensors is not exploited. In this chapter, we introduced the third module, which fills this gap and refines the localization estimate.

The refinement module uses the estimation from the coarse localization. Since the coarse localization narrows down the feasible set of poses based on the always valid constraints, the refinement module performs a points-to-facade association of the LiDAR points. By considering the LiDAR measurement uncertainty and the building map uncertainty, our refinement module further narrows down the feasible region to a smaller polygon for the position and a smaller orientation interval by exploiting the associations. However, the refinement can only be performed if the feasible set provided by the coarse localization is accurate enough to perform unambiguous matching. Hence, the refinement has to wait until the coarse localization result is accurate enough to operate.

Note that the consistent set is not necessarily reliable. In the case of an incorrect association, the result provided by the refinement may be error-prone. Nonetheless, if the association is correct, the refinement significantly improves the accuracy of the localization estimate. To improve the overall localization performance, in the case of correct associations, we can use the refined results to improve the coarse localization to accelerate convergence and gain computation time. However, the main problem here is that we must ensure that the current track considers correct associations. The next chapter introduces our solution to this problem and provides an overview of how the modules are connected to the HyPaSCoRe Localization system.

7 HyPaSCoRe Localization Pipeline

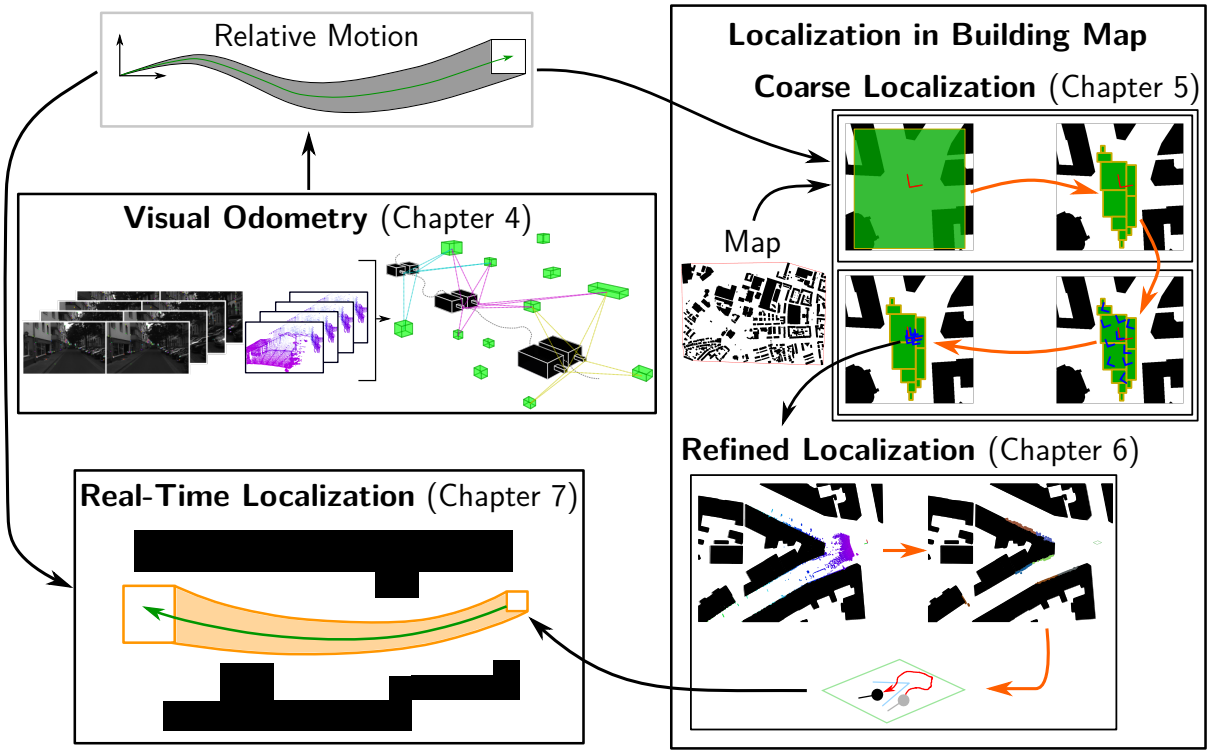


Figure 7.1: HyPaSCoRe Localization overview.

Up to this point, we introduced the different components of the HyPaSCoRe Localization pipeline. This chapter aims to bring all components together into one localization application and highlight the interfaces and collaboration between the modules. Furthermore, we will focus on the localization estimation in real-time. Figure 7.1 shows the complete schematic overview of the HyPaSCoRe Localization pipeline.

The visual odometry module uses stereo images and LiDAR data to compute the relative motion of the vehicle. The visual odometry module consists of the front-end and the back-end. In the front-end, visual features are extracted from the stereo images, and the depth information from the LiDAR sensor further augments the depth estimation of the stereo feature. As features are tracked along the trajectory, the dependencies between the poses and the landmarks are stored in a SLAM-Graph. Furthermore, within the front-end, we determine the frame-to-frame relative motion that initializes the windowed bundle adjustment in the back-end. The robust

windowed bundle adjustment is applied on a sequence of poses during which the same landmarks are observed. Using the bundle adjustment, we determine outliers and exclude them from the interval-based trajectory estimation.

The coarse localization uses the relative motion obtained by the visual odometry to localize the vehicle inside the building map. The coarse localization consists of two steps. First, using interval analysis, the feasible set of poses under consideration of the local LiDAR data, the visual odometry, and occasionally available uncertain GNSS measurements is narrowed down to a smaller set. Second, a bounded MCL is performed within the feasible set to determine the most likely solution. However, the feasible set is typically large as the coarse localization only uses always valid but less restrictive constraints to contract the feasible region.

To improve the localization estimate, we utilize the refined localization. It refines the coarse localization result by introducing point-to-facade associations. Using the particle with the highest weight from bounded MCL of the coarse localization, local LiDAR points are associated with facades in the building map in a nearest-neighbor fashion. The associations make further contraction of the feasible set from the coarse localization possible as we narrow down the set of feasible poses to the subset consistent with the point-to-facade associations. Furthermore, we use the associations to determine the most likely pose within the consistent set by applying a bounded least squares optimization.

To ensure the real-time capability of our method, a multi-threaded architecture of the localization pipeline is mandatory that exploits independent processes to speed up the method. In Section 7.1, we first present the general architecture of our implementation of the localization pipeline. The localization process is always delayed by the bundle adjustment in the visual odometry module, so we cannot directly determine the location of the current pose. Therefore, we introduce a workaround in Section 7.2 that provides the most likely pose and the uncertainty bound for the current frame making real-time localization possible.

Our HyPaSCoRe Localization pipeline does not only perform a straightforward localization from coarse to refined localization. To improve the performance of the whole localization pipeline, we also introduce feedback from the refined localization to the coarse localization to control the exploration region to accelerate the convergence speed, which is described in Subsection 7.3.2.

7.1 Architecture

The general software architecture of our localization pipeline is displayed in Figure 7.2. The visual odometry uses two parallel threads. The first thread runs the front-end that constructs the SLAM-Graph. This thread has to process the sensor data with the sensor data rate for real-time capability. Based on the received sensor data, the front-end extracts landmarks and stores the observations to the SLAM-Graph as explained in Section 4.2. Hence, the SLAM-Graph is the data structure that connects the front- and the back-end. The second thread executes the back-end. As presented in Section 4.3, the back-end solves the SLAM-Graph by applying a windowed bundle adjustment and the interval-based visual odometry. However, this process runs on a lower frequency than the data rate since the back-end is only initiated if a keyframe is inserted. As the back-end processes the whole batch considering multiple frames, the back-end

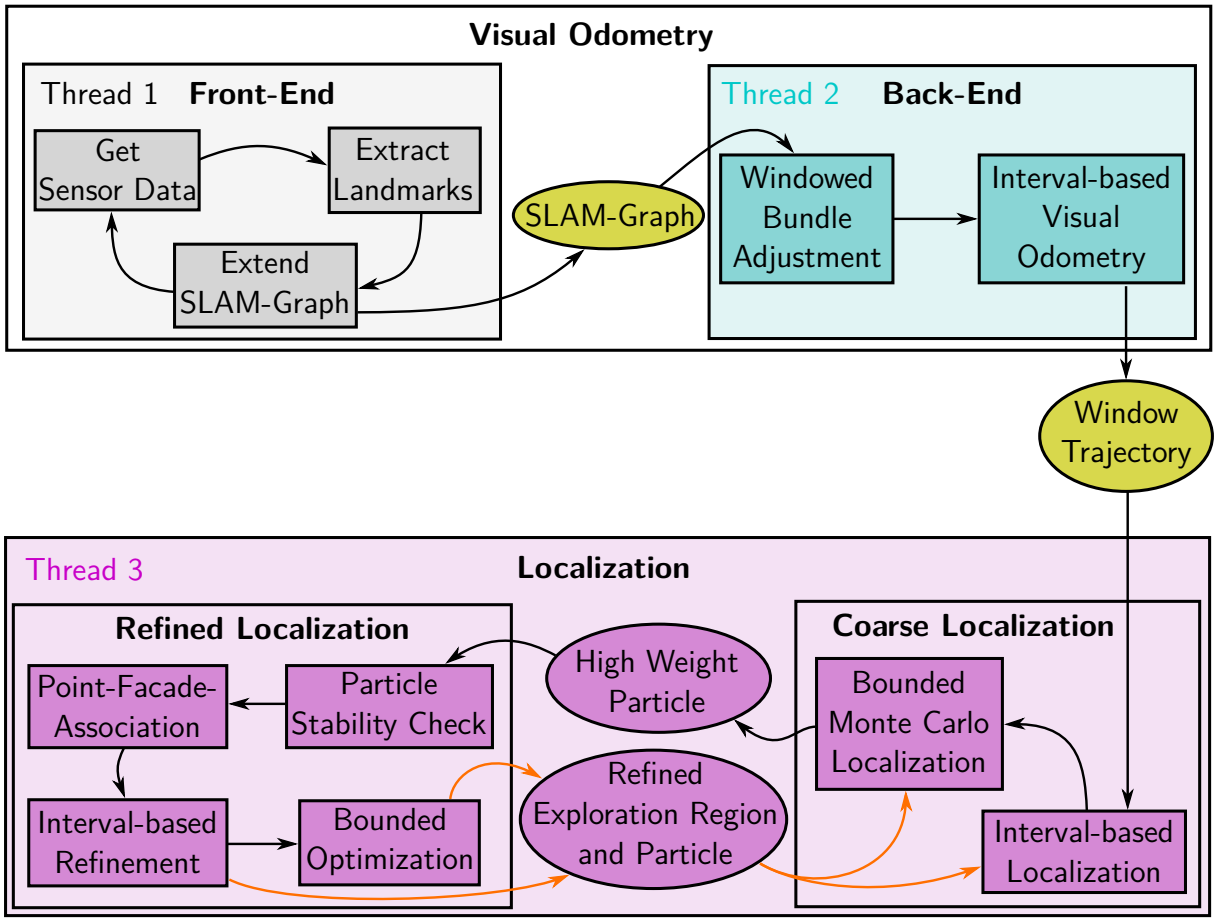


Figure 7.2: Software architecture of the HyPaSCoRe Localization pipeline. The pipeline uses three host threads. Two threads are dedicated to the visual odometry, while the localization uses one host thread. The different tasks that need to be solved by the processes are bounded by boxes and are colored according to the thread to which they are assigned. Ellipses bound messages between tasks/processes. Messages between two threads are colored yellow.

does not need to be executed at sensor frequency. Nonetheless, the process needs to be finished until the next trajectory window is closed by a new keyframe.

Once the back-end thread of the visual odometry has processed the batch, the localization is triggered. Consequently, we process the whole window of frames sequentially at once. Thus, the localization procedure is (at maximum) delayed by the window size of a batch. Since the localization is executed faster per frame than the sensor data rate, the localization procedure quickly keeps up with the current frame as soon as a new batch of frames is ready for localization. Nonetheless, with this architecture, real-time localization is not possible. We deduce the real-time localization to the current frame at the data rate by considering the less accurate frame-to-frame odometry computed in the front-end based on the recent localization estimate. The real-time localization is presented in more detail in Section 7.2.

The localization is executed on a single host thread as the tasks are performed sequentially. The localization processes the trajectory window frame by frame. First, the coarse localization begins with interval-based localization (cf. Section 5.2). Based on the feasible set, a bounded MCL is performed to obtain the most likely poses as presented in Section 5.3.

The refined localization is only executed if the particles obtained from the coarse localization are staged as stable. The stability check will be presented in more detail in Subsection 7.3.1. The refined localization performs the point-to-facade association and computes the consistent polygon set and orientation interval in the interval-based refinement. The newly obtained bounds are used for the bounded optimization to obtain the most likely pose within the consistent set.

As stated previously, the refined localization provides more accurate localization results than the coarse localization. The higher accuracy comes with the cost of less integrity due to the possibly error-prone association step. However, the localization results are more accurate in the case of a correct association since the consistent set represents a tighter bound. Hence, if the association is correct, we can improve the coarse localization by contracting the feasible set to the consistent set. The exploration region can be further contracted, accelerating the convergence. The idea is to take advantage of the higher accuracy and only consider the smaller set if we can be sure the refined estimate is trustworthy. Subsection 7.3.2 presents how we do that. With this approach, we maintain high integrity standards and accuracy for the localization result. This feedback mechanism from the refined localization to the coarse localization is indicated by the orange arrows Figure 7.2.

7.2 Real-Time Localization

The localization is only triggered when the back-end of the visual odometry has finished processing the current window. Consequently, the localization is always delayed as the back-end only processes closed windows. That means our architecture cannot provide the localization estimate for the sensor frame to date. Nonetheless, based on the recent localization estimate, we predict the current pose to frame rate. Therefore, we utilize the real-time frame-to-frame relative pose estimate computed in the front-end of the visual odometry as presented in Subsection 4.2.4.

Let ${}^{M}\hat{\mathbf{T}}_{L_{t-k}}$ be the pose parameters of the frame that is processed at the timestep t when the newest data from the sensors are received. Due to the delay in the visual odometry back-end, the processed pose corresponds to the data frame $\mathcal{F}_{t-k}$ at time $t-k$. As a result, the localization is delayed by k timesteps. The refined localization provides the consistent set of poses ${}^{M}\mathcal{P}^{\triangle}_{L_{t-k}} = \left({}^{M}\mathcal{T}^{\triangle}_{L_{t-k}}, [{}^{M}\varphi^{\triangle}_{L_{t-k}}]\right)^{T}$ described by a polygon ${}^{M}\mathcal{T}^{\triangle}_{L_{t-k}}$ for the translation and an orientation interval $[{}^{M}\varphi^{\triangle}_{L_{t-k}}]$, and the most likely pose ${}^{M}\hat{\mathbf{T}}^{*}_{L_{t-k}}$ within the consistent set. The coarse localization provides the feasible set of poses ${}^{M}\mathcal{P}^{\boxplus}_{L_{t-k}} = \left({}^{M}\mathcal{T}^{\boxplus}_{L_{t-k}}, {}^{M}\Phi^{\boxplus}_{L_{t-k}}\right)^{T}$ with a 2D subpaving for the translation ${}^{M}\mathcal{T}^{\boxplus}_{L_{t-k}}$ and a 1D subpaving ${}^{M}\Phi^{\boxplus}_{L_{t-k}}$ for the orientation interval. As those estimates correspond to $\mathcal{F}_{t-k}$ at time $t-k$ the goal is to determine the current pose ${}^{M}\hat{\mathbf{T}}_{L_t}$ and the set-membership estimates ${}^{M}\mathcal{P}^{\triangle}_{L_t}$ and ${}^{M}\mathcal{P}^{\boxplus}_{L_t}$ at time t. The problem is that at time t, we do not have reliable relative pose estimates for all frames between t and $t-k$ as illustrated in Figure 7.3. The windowed bundle adjustment was not processed for the last frames, as the current window has not been closed.

Fortunately, in the front-end of the visual odometry, we determine frame-to-frame odometry by applying a classical LM optimization that minimizes the reprojection error of the commonly

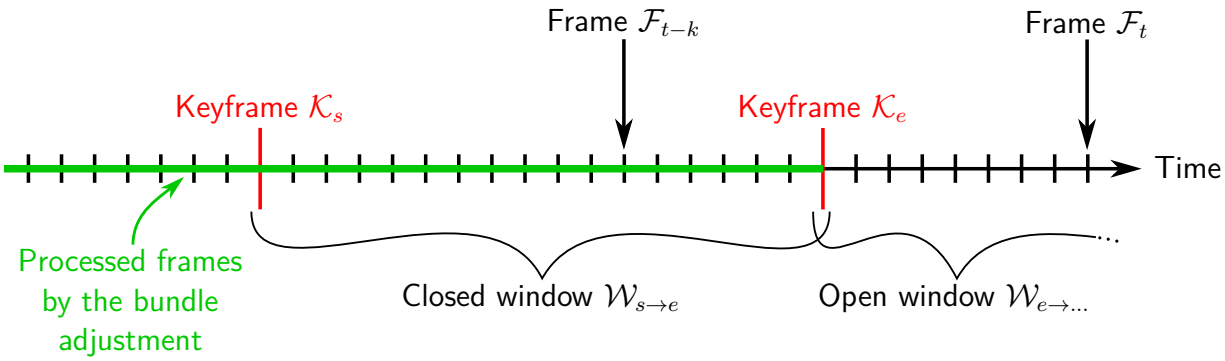


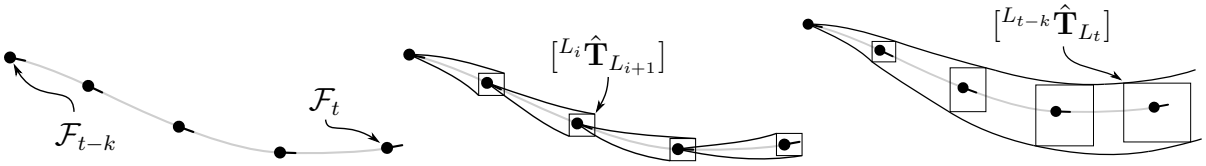
Figure 7.3: Timeline with processed and unprocessed frames. The frames highlighted in red are the keyframes $\mathcal{K}_s$ and $\mathcal{K}_e$ that define the closed window $\mathcal{W}_{s\to e}$. Since the windowed bundle adjustment is only initiated if a window of frames is closed by inserting a keyframe, the frames after $\mathcal{K}_e$ are not yet processed by the bundle adjustment. That is why we do not have accurate relative pose information for those frames.

observed stereo features of consecutive frames in real-time. Hence, we first estimate the relative transformations between the k consecutive poses between timesteps $t-k$ and t. However, those estimates are not necessarily reliable. Formerly, we used the estimates in the back-end only to initialize the windowed bundle adjustment that improves the poses in the batch since dynamic features may corrupt the frame-to-frame estimates. Nonetheless, we can use the relative transformations to predict the vehicle's pose at time t by considering a comparatively large uncertainty for the estimate. Note that the frame-to-frame pose estimation in the front-end of the visual odometry is an MLE approach that only provides a scalar-valued estimate on the pose parameters ${}^{L_i}\hat{\mathbf{T}}_{L_{i+1}}$ for an arbitrary relative pose between frames at i and $i+1$. Hence, we do not have a direct estimate of the uncertainty of the relative transformation. That is why we inflate the scalar-valued relative pose estimates to intervals. Therefore, we account for a comparatively large uncertainty to compensate for the possibly error-prone relative pose estimate. The width by which we inflate the pose estimates is fixed for the parameters, and we determined the parameters empirically. For the translation, we choose $0.005\,\mathrm{m}$, and for rotation, 1° inflation radius.

As we want to determine the pose ${}^{M}\hat{\mathbf{T}}_{L_t}$ at time t in the map frame and the pose ${}^{M}\hat{\mathbf{T}}_{L_{t-k}}$ at $t-k$ is known, we determine the relative pose ${}^{L_{t-k}}\hat{\mathbf{T}}_{L_t}$ by concatenating the relative transformations between the consecutive poses between the time steps $t-k$ and t. Therefore, we represent the transformation parameters ${}^{L_{t-i}}\hat{\mathbf{T}}_{L_t}$ for $i \in \{0, \dots, k\}$ by the affine transformation ${}^{L_{t-i}}\mathbf{T}_{L_t}$ and perform the concatenation by

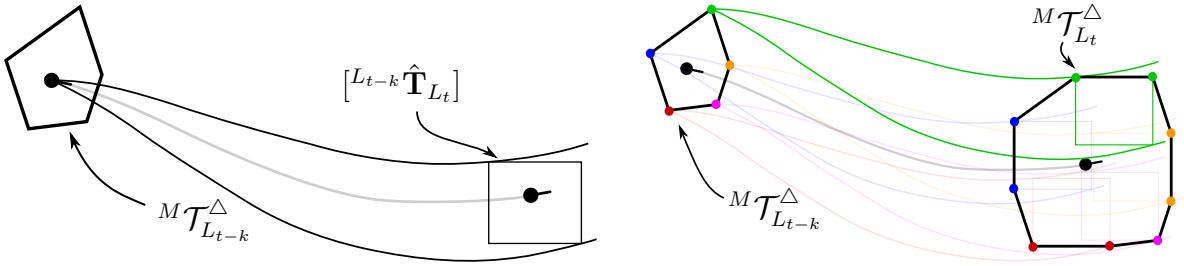
$$ {}^{L_{t-k}}\mathbf{T}_{L_t} = {}^{L_{t-k}}\mathbf{T}_{L_t} \cdot {}^{L_{t-(k-1)}}\mathbf{T}_{L_t} \cdot \dots \cdot {}^{L_{t-1}}\mathbf{T}_{L_t}. \tag{7.1} $$

Since the relative poses ${}^{L_{t-i}}\hat{\mathbf{T}}_{L_t}$ are represented by intervals as we inflate the frame-to-frame pose estimates, the concatenation of the transformations are extended to interval operations so that ${}^{L_{t-i}}\hat{\mathbf{T}}_{L_t} \in {}^{L_{t-i}}\mathcal{P}_{L_t}$. As a predefined uncertainty inflates each relative pose, the concatenation of the transformation accordingly accumulates the uncertainties as illustrated in Figure 7.4. The obtained relative pose is used to determine the current pose ${}^{M}\hat{\mathbf{T}}_{L_t}$ at t by transforming the ${}^{M}\hat{\mathbf{T}}_{L_{t-k}}$ with ${}^{L_{t-k}}\hat{\mathbf{T}}_{L_t}$.



(a) Ground truth trajectory. (b) Relative pose uncertainty. (c) Accumulated pose uncertainty.

Figure 7.4: Accumulation of uncertain relative transformations. While Figure 7.4a shows a schematic ground truth trajectory, Figure 7.4b visualizes the interval-based relative pose estimate at different timesteps. The concatenation of the relative transformations accumulates the uncertainty using interval tools. The result is visualized in Figure 7.4c.



(a) Set of consistent poses at time $t-k$ and the relative transform for a selected point. (b) Transformation of the consistent set described by a polygon by the relative transform.

Figure 7.5: Relative transformation of a polygon. Figure 7.5a shows the position estimate ${}^{M}\mathcal{T}^{\triangle}_{L_{t-k}}$ of the refined localization at time $t-k$. Furthermore, the relative transform from Figure 7.4c is shown for an exemplary point inside ${}^{M}\mathcal{T}^{\triangle}_{L_{t-k}}$. To determine ${}^{M}\mathcal{T}^{\triangle}_{L_t}$, we transform each corner point of ${}^{M}\mathcal{T}^{\triangle}_{L_{t-k}}$ by the relative transform $[{}^{L_{t-k}}\hat{\mathbf{T}}_{L_t}]$ so that we obtain multiple boxes as illustrated in Figure 7.5b. We define the ${}^{M}\mathcal{T}^{\triangle}_{L_t}$ as the convex polygon hull over all generated boxes.

First, let us consider the transformation of the coarse localization estimate ${}^{M}\mathcal{P}^{\boxplus}_{L_{t-k}}$. However, transforming a subpaving can become heavy in computation since the transformation has to be applied to each subset. That is why we approximate the subpaving by a box hull – a hull over the translation subpaving and a hull over the orientation subpaving. Consequently, ${}^{M}\mathcal{P}^{\boxplus}_{L_{t-k}}$ simplifies to ${}^{M}\mathcal{P}^{\square}_{L_{t-k}}$ with three intervals, where ${}^{M}\mathcal{T}^{\square}_{L_{t-k}}$ is a simple 2D box and ${}^{M}\Phi^{\square}_{L_{t-k}}$ is an interval. The translation is determined by

$$ {}^{M}\mathbf{t}_{L_t} = \begin{pmatrix} \cos {}^{M}\varphi_{L_{t-k}} & -\sin {}^{M}\varphi_{L_{t-k}} \\ \sin {}^{M}\varphi_{L_{t-k}} & \cos {}^{M}\varphi_{L_{t-k}} \end{pmatrix} .{}^{L_{t-k}}\mathbf{t}_{L_t} +^{M} \mathbf{t}_{L_{t-k}}, \tag{7.2} $$

for ${}^{M}\mathbf{t}_{L_{t-k}} \in^{M} \mathcal{T}^{\square}_{L_{t-k}}$, ${}^{M}\varphi_{L_{t-k}} \in^{M} \Phi^{\square}_{L_{t-k}}$ and ${}^{L_{t-k}}\mathbf{t}_{L_t} \in^{L_{t-k}} \mathcal{T}_{L_t}$. The rotation is determined by ${}^{M}\varphi_{L_t} =^{M} \varphi_{L_{t-k}} +^{L_{t-k}} \varphi_{L_t}$ for ${}^{M}\varphi_{L_t} \in^{M} \Phi_{L_t}$, ${}^{M}\varphi_{L_{t-k}} \in^{M} \Phi^{\square}_{L_{t-k}}$ and ${}^{L_{t-k}}\varphi_{L_t} \in^{L_{t-k}} \Phi_{L_t}$.

Now, let us consider the refined localization estimate that provides a polygon ${}^{M}\mathcal{T}^{\triangle}_{L_{t-k}}$ for the position and an interval $[{}^{M}\varphi^{\triangle}_{L_{t-k}}]$ for the orientation. To transform the polygon and the orientation interval from $t-k$ to t, we can use SIVIA. However, as we want to perform the transformation in real-time, a branch-and-bound algorithm may exceed the time budget for special constellations, leading to multiple bisections. That is why we propose an approximation.

The polygon ${}^{M}\mathcal{T}^{\triangle}_{L_{t-k}}$ is defined by multiple corner points that define the border of the polygon. We consider each corner point as a feasible position for the vehicle. Hence, we transform each corner pose by the concatenated relative pose as shown in (7.2). Hence, we obtain a position box for each corner pose at $t-k$ and an orientation interval at time t. We define the feasible set of positions at time t by the convex polygon hull among all position boxes generated at t by the corner poses. The orientation is defined by the orientation hull among all corner poses. Figure 7.5 illustrates our approach. The most likely pose is determined by classically transforming the most likely pose obtained by the refined localization by the bounded optimization.

7.3 Cooperations between the Coarse and Refined Localization

The refined localization depends on the coarse localization results. Since the refined localization performs a point-to-facade association, we need a comparatively accurate result from the coarse localization to initialize the refined localization. Up to this point, only the coarse localization is processed. How do we detect that a result is accurate enough to initiate the refined localization? Therefore, we introduce the particle stability check in Section 7.3.1.

After the refined localization has been initiated, we will obtain two different estimates of the vehicle location. While the coarse localization provides the feasible set of poses as a subpaving ${}^{M}\mathcal{P}^{\boxplus}_{L_t}$, the refined localization provides a polygon set for the translation and an interval for the rotation that we describe by ${}^{M}\mathcal{P}^{\triangle}_{L_t}$. The set ${}^{M}\mathcal{P}^{\boxplus}_{L_t}$ obtained by the coarse localization is very reliable to contain the correct pose since the estimation is based on always valid and simple constraints. On the downside, ${}^{M}\mathcal{P}^{\boxplus}_{L_t}$ is large. In contrast, the refined localization estimate ${}^{M}\mathcal{P}^{\triangle}_{L_t}$ provides less pessimistic results. However, the estimate is less reliable since it uses a data associations step that may be error-prone. Hence, how can we fuse both estimates to obtain the best of both estimates and have reliable and accurate location estimates? In Section 7.3.2, we present our approach to obtain a reliable and accurate estimate based on ${}^{M}\mathcal{P}^{\boxplus}_{L_t}$ and ${}^{M}\mathcal{P}^{\triangle}_{L_t}$.

7.3.1 Particle Stability Check

Based on the coarse localization results, the particle stability check determines when to initiate the refined localization. Therefore we exploit the stability of the particle-based pose estimate of the bounded MCL: Initially, particles are very unlikely to be spread close to the correct pose. Since we perform aggressive resampling with the random spreading of particles, only high-weighted particles will survive because inconsistent particles will be replaced. As a result, the age of a particle, which indicates how many iterations it survived, can be chosen as a metric of how reliable the particle estimate is. The longer a particle survives, the more reliable it is, as it has proven to be consistent for at least a part of the trajectory.

Note that a particle resampled from another high-weighted particle inherits the age. Consequently, in the case of convergence to a consistent pose, the bounded MCL will form a cluster of particles with the same comparatively high age. The age of a particle is a metric for measuring the consistency of the local measurements to the map location. The consistency

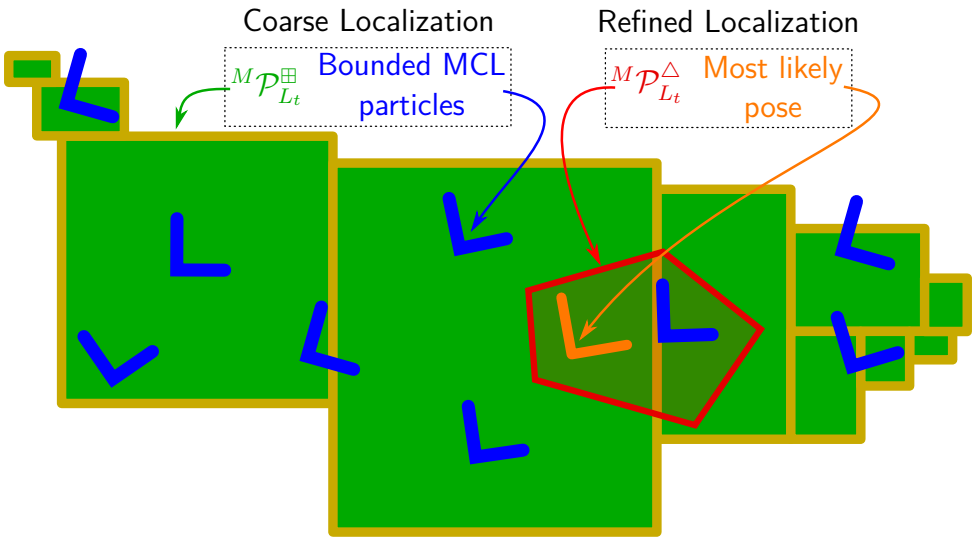


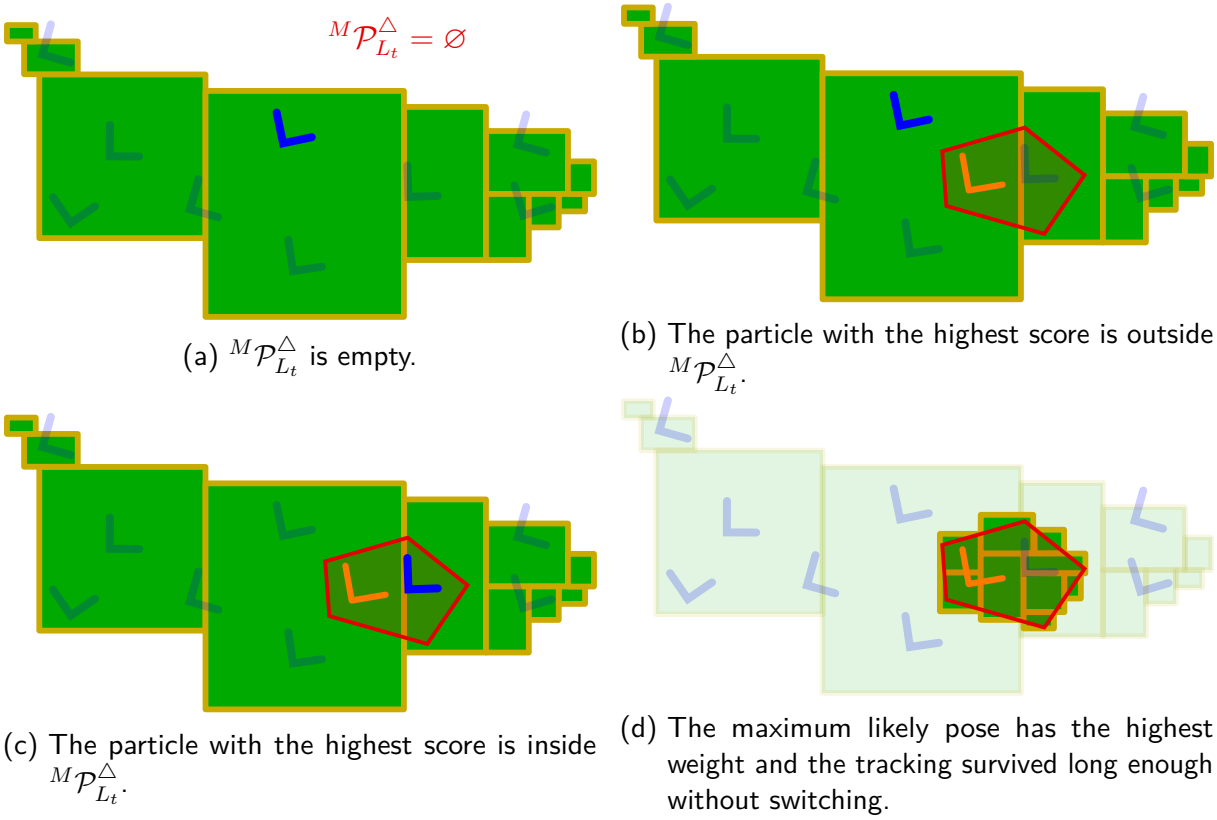
Figure 7.6: The coarse and refined localization results. The coarse localization describes the feasible set of locations of the vehicle by a subpaving $^{M}\mathcal{P}^{\boxplus}_{L_t}$ and a set of weighted particles of the bounded MCL. The refined localization provides a set of poses $^{M}\mathcal{P}^{\triangle}_{L_t}$ where the translation set is described by a polygon and the orientation by an interval and a most likely pose from the MLE approach. Both approaches provide a set-membership-based and probabilistic estimate of the vehicle location.

indicates that the bounded MCL tracks the correct pose. However, we cannot guarantee that the consistent pose is correct. For instance, in the case of symmetries in the map, multiple consistent poses may exist as the local measurements can be consistently matched at different locations. In that case, we do not have any possibility to distinguish if a tracked consistent pose is indeed the correct one. However, this corner case is not a specific problem of our approach. Also, as presented in Chapter 3, State of the Art approaches have difficulties in such corner cases. Although the bounded MCL may suffer from symmetries in the map, the interval-based approach in the coarse localization counteracts this problem as it will still identify all feasible locations of the vehicle correctly and will be able to deal with symmetries in contrast to other State of the Art methods.

If the age of a particle exceeds a given threshold, we initiate the refined localization. To initialize the refined localization, we consider the particle with the highest weight to determine the point-to-facade association. Based on the association, the refined localization provides a minimal polygon and a smaller interval for the orientation than the coarse localization.

7.3.2 Bound the Exploration Region – Reliability of the Refined Localization

After the initiation of the refined localization, our pipeline provides two different estimates of the vehicle location. The coarse localization provides a comparatively pessimistic but very reliable subpaving $^{M}\mathcal{P}^{\boxplus}_{L_t}$ and a set of particles. The refined localization provides a polygon pose set $^{M}\mathcal{P}^{\triangle}_{L_t}$ that is less pessimistic but also less reliable. Furthermore, the refined localization provides the most likely pose within $^{M}\mathcal{P}^{\triangle}_{L_t}$ that best satisfies the association. Figure 7.6 illustrates the localization results we obtain from both approaches. The main problem of $^{M}\mathcal{P}^{\triangle}_{L_t}$



(a) ${}^{M}\mathcal{P}^{\triangle}_{L_t}$ is empty.

(b) The particle with the highest score is outside ${}^{M}\mathcal{P}^{\triangle}_{L_t}$.

(c) The particle with the highest score is inside ${}^{M}\mathcal{P}^{\triangle}_{L_t}$.

(d) The maximum likely pose has the highest weight and the tracking survived long enough without switching.

Figure 7.7: Different scenarios for track switching. Figure 7.7a directly leads to a full reset as ${}^{M}\mathcal{P}^{\triangle}_{L_t} = \varnothing$ indicates that the association is inconsistent. In Figure 7.7b, the particle outside the consistent set has a higher weight than the tracked pose. If the blue particle has survived a minimal number of iterations, the track is switched to the blue particle and is fully reset. In Figure 7.7c, the blue particle with higher weight is inside ${}^{M}\mathcal{P}^{\triangle}_{L_t}$. If the blue particle is old enough, the track is internally switched. If the tracked pose (orange) has the highest weight and survives a minimal number of iterations, ${}^{M}\mathcal{P}^{\boxplus}_{L_t}$ is contracted to ${}^{M}\mathcal{P}^{\triangle}_{L_t}$ as shown in Figure 7.7d.

is that it relies on the point-to-facade association that can be error-prone. If the association is correct, it is guaranteed to contain the correct pose. However, we cannot assure that the association is correct, and consequently, we cannot guarantee that ${}^{M}\mathcal{P}^{\triangle}_{L_t}$ encloses the correct pose.

However, we can use the same consistency metric that we introduce in Subsection 7.3.1 to measure the trustworthiness of the refined localization result. When the refined localization is initiated, a track is initiated by the particle with the highest weight from the bounded MCL of the coarse localization. We define the trustworthiness of a track based on the number of timesteps the track survives, similar to the particle stability in Subsection 7.3.1. Track switches for which we reset the age are performed in the following cases:

- Either the rotation interval or the translation polygon of ${}^{M}\mathcal{P}^{\triangle}_{L_t}$ is empty as illustrated in Figure 7.7a. The empty set indicates that the association in the refined localization is inconsistent. Hence, the current track has to be incorrect.
- The coarse localization tracks multiple particles, while the refined localization represents the most likely solution by one tracked particle. Suppose there is a better particle

that is old enough and has a higher score than the currently tracked particle in the refined localization. In that case, the track is switched to the higher-weighted particle from the coarse localization. This scenario is illustrated in Figure 7.7b and Figure 7.7c. To determine the score of a particle, we utilize the beam-end model as presented in Section 5.3.

When necessary, the track is switched to the particle with the highest weight.

We distinguish two types of track switches. The first type considers a full reset of the refined localization, equivalent to reinitializing the whole refined localization. When a full reset is necessary, the prediction polygon for the next iteration step is not determined by the odometry update as presented in Section 6.2.1 for the usual case. In contrast, the prediction polygon is defined by the hull of the interval-based estimate ${}^{M}\mathcal{P}_{L_t}^{\boxplus}$ of the coarse localization so that the wrong track does not affect the following iterations in the refined localization. Furthermore, the age of the track is set to zero. The full reset is applied in two cases which are visualized in Figure 7.7a and Figure 7.7b: First, if ${}^{M}\mathcal{P}_{L_t}^{\triangle}$ is empty or second, if there is a better particle that is outside ${}^{M}\mathcal{P}_{L_t}^{\triangle}$. In both cases, the association provides inconsistent or less consistent results than alternative solutions tracked by the bounded MCL in the coarse localization.

The second type considers an internal switch of the refined localization when a full reset is unnecessary. The switch is applied for the case when there is a better particle inside ${}^{M}\mathcal{P}_{L_t}^{\triangle}$ as illustrated in Figure 7.7c. As the better particle is compatible with the association of the current track, the polygon is used to predict the pose for the next iteration step. Only the age of the track is set to zero, and the tracked particle is switched to the better particle provided by the coarse localization. As a consequence, the main difference between the full reset and the internal switch is that the tracked set ${}^{M}\mathcal{P}_{L_t}^{\triangle}$ is treated differently for the next iteration step.

If a track survives long enough and exceeds a defined number of iterations, the track is classified as reliable. In that case, we contract the feasible region ${}^{M}\mathcal{P}_{L_t}^{\boxplus}$ to ${}^{M}\mathcal{P}_{L_t}^{\triangle}$. This contracts the exploration region of the coarse localization as shown in Figure 7.7d. We only perform the contraction if differently oriented facades are observed. This accelerates the convergence of the localization as the bounded MCL concentrates on a smaller set and reduces the runtime because the subpaving shrinks. Furthermore, we correct the highest particle pose of the coarse localization by the most likely pose estimate from the refined localization to further consider the refined localization result in the next iteration of the bounded MCL in the coarse localization.

8 Experimental Results

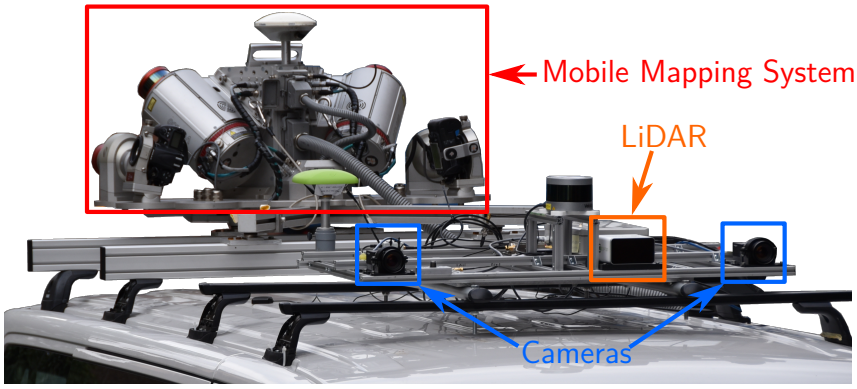
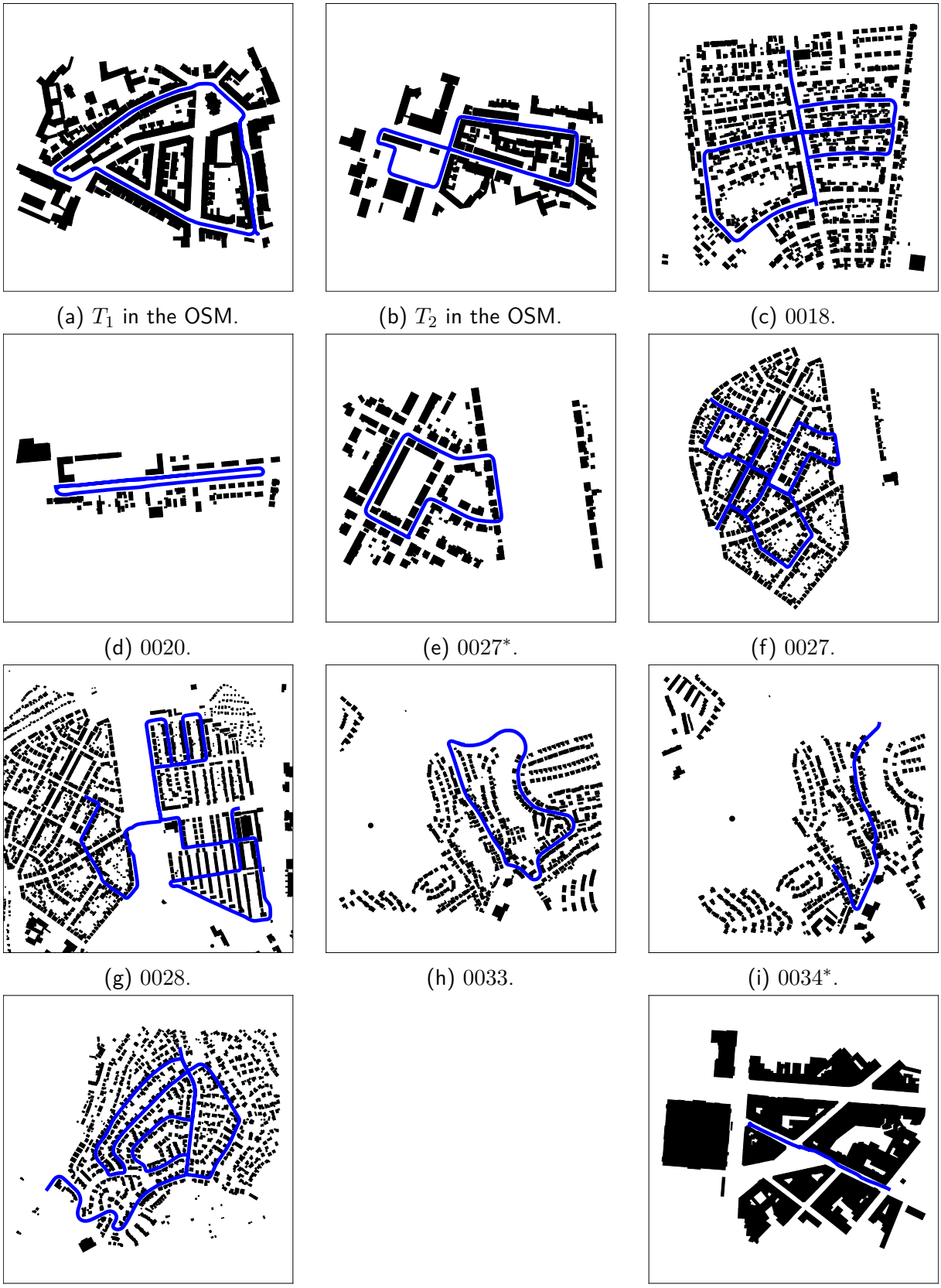


Figure 8.1: The sensor platform mounted on the test vehicle acquired the author-collected datasets.

We evaluate the HyPaSCoRe Localization with eleven different datasets. Two are author-collected datasets (T_1 and T_2), while nine are from the KITTI benchmark [146]. In the KITTI datasets, we use the images provided by the two PointGray Flea2 color cameras (FL2-14S3C-C) with an image resolution of 1241×376 pixels. The cameras are equipped with Edmund Optics lenses, and the base length of the stereo setup measures $0.54\,\mathrm{m}$. Furthermore, the LiDAR data comes from a Velodyne HDL-64E rotating 3D laser scanner. We utilize the OXTS RT3003 inertial and GPS navigation system measurements as a ground truth reference for the poses. The whole setup of the test vehicle can be found in [146].

We acquired our datasets in an urban region in Hanover, Germany. The sensor setup we mounted on our test vehicle is presented in Figure 8.1. We use two FLIR Grasshopper3 color cameras (GS-U3-23S6C-C) with a 1920×1200 pixels resolution. Each of the cameras is equipped with a Fujinon CF12.5HA-1 lens. The stereo baseline measures a length of $0.85\,\mathrm{m}$. In contrast to the KITTI dataset, we use the MEMS solid-state LiDAR Cepton Vista 8800. The ground truth measurements are acquired with the Riegl VMX-250 Mobile Mapping System.

We visualize the maps and the ground truth trajectories in Figure 8.2. Note that we selected different map topologies with different building densities to evaluate our HyPaSCoRe Localization method. We summarize further information on the datasets in Table 8.1. While we test all trajectories with the OSM data downloaded from the database [70], the LOD2 map is only tested with the author-collected datasets T_1 and T_2. Note that the KITTI trajectories 0027^* and 0034^* were acquired on the 30th of September 2011. We put a star in the index



(a) T_1 in the OSM. (b) T_2 in the OSM. (c) 0018.

(d) 0020. (e) 0027*. (f) 0027.

(g) 0028. (h) 0033. (i) 0034*.

(j) 0034. (k) 0071.

Figure 8.2: Trajectories and OSM building data for all datasets. The trajectories are colored blue.

Dataset	Length	Frames	Map size	
			OSM	LOD2
T_1	1249 m	2866	365 m × 479 m	1036 m × 2050 m
T_2	1054 m	2795	291 m × 512 m	1036 m × 2050 m
0018	2206 m	2750	593 m × 632 m	-
0020	1230 m	1096	175 m × 580 m	-
0027*	684 m	1070	300 m × 436 m	-
0027	3081 m	3824	895 m × 717 m	-
0028	3230 m	4086	1966 m × 2840 m	-
0033	1706 m	1593	852 m × 998 m	-
0034*	897 m	1179	955 m × 935 m	-
0034	5062 m	4634	1081 m × 1263 m	-
0071	239 m	1058	327 m × 394 m	-

Table 8.1: Datasets. The trajectory length, the number of LiDAR frames, the number of buildings in the map, and the map size of the different datasets are listed. The size is measured by the length in the north direction times the length in the east direction.

to distinguish them from the other trajectories to maintain unique naming. For naming the KITTI trajectories, we follow the raw-data naming on the KITTI web page [146].

With the KITTI datasets, we experience problems with the ground truth trajectory. The ground truth data provided by the KITTI benchmark has occasional offsets to the OSM data. In rare cases, there are also jumps in the ground truth trajectory. We visualize such an exemplary jump in 0028 in Figure 8.3. Although the error in 0028 is an extreme example that occurs very rarely, during the evaluation, we need to remember that the KITTI ground truth needs to be considered cautiously as it is unreliable. In contrast, the ground truth of the author-collected datasets has centimeter accuracy and is, therefore, very reliable.

All experiments are performed on one consumer-grade laptop equipped with an Intel Core i7-9850H @ 2.6 GHz × 12 CPU, 64 GB RAM and an NVIDIA Quadro T2000 GPU. We perform the image processing and feature extraction in the front-end of the visual odometry module on the GPU. However, all other modules are processed on the CPU.

8.1 Visual Odometry

In this section, we evaluate the first module of our localization pipeline. The visual odometry module uses the stereo images and the LiDAR data to determine the local frame-to-frame motion of the vehicle as presented in Chapter 4. Therefore, we detect and track visual features as landmarks. Since we use a SLAM-graph architecture to determine the vehicle's trajectory, we can divide the visual odometry into two submodules that we evaluate separately.

The first submodule is the front-end which builds the graph based on the sensor data. Consequently, in Subsection 8.1.1, we focus our analysis of the experimental results on feature detection and tracking. Since we implement a keyframe-based approach, we batch the incoming frames into multiple windows for which we construct the graphs.

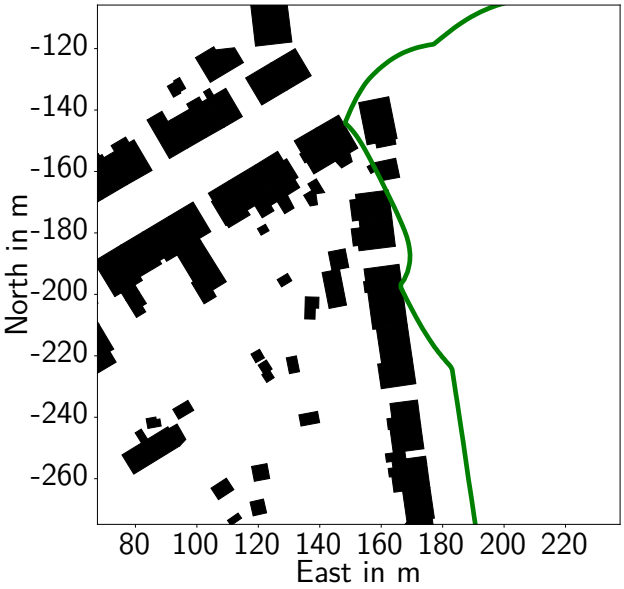


Figure 8.3: Exemplary error of the ground truth trajectory in 0028.

The second submodule is the back-end which processes the SLAM-graph built by the front-end to determine, on the one hand, outliers of landmark observations and, on the other hand, the relative motion between the consecutive frames in the window that is processed. Hence, in Subsection 8.1.2, we evaluate the reprojection errors of the visual features, the detected outliers, and the reconstructed landmark uncertainties after applying the windowed bundle adjustment and the interval-based odometry estimation. Furthermore, we provide insights into the uncertainty of the interval odometry and compare the results to a classical MLE approach for a small exemplary trajectory. Finally, we provide the runtime evaluation in Subsection 8.1.3.

8.1.1 Front-End Evaluation

The front-end detects and tracks visual features in the stereo images. Hence, depending on how well and how many features are tracked in the images, the front-end determines the length of a window processed in the back-end in the windowed bundle adjustment. The longer a window is, the longer features are tracked, which means we acquire more information on a potential landmark. The longer windows are, the better it is for the windowed bundle adjustment since more information is processed once in the optimization, and outliers are more likely to be detected. However, as the vehicle is driving and the sensors capture new parts of the scene, a window cannot have an infinite length. As described in Subsection 4.2.1, we introduce a new window when the number of tracked features drops below a predefined threshold. In the scope of the experiments, we set this threshold to 80 based on empirical studies.

To evaluate the length of the windows and how long landmarks are tracked along the trajectory, we provide two exemplary histograms. While Figure 8.4 depicts the window lengths in T_1, Figure 8.5 shows the tracking length of all valid landmarks. In Table 8.2, we provide results with condensed evaluation metrics for all trajectories.

The histogram in Figure 8.4 shows that most windows in T_1 have a length below $5\,\mathrm{m}$. Nonetheless, some windows are longer than $10\,\mathrm{m}$. The longest window has a length of $33\,\mathrm{m}$.

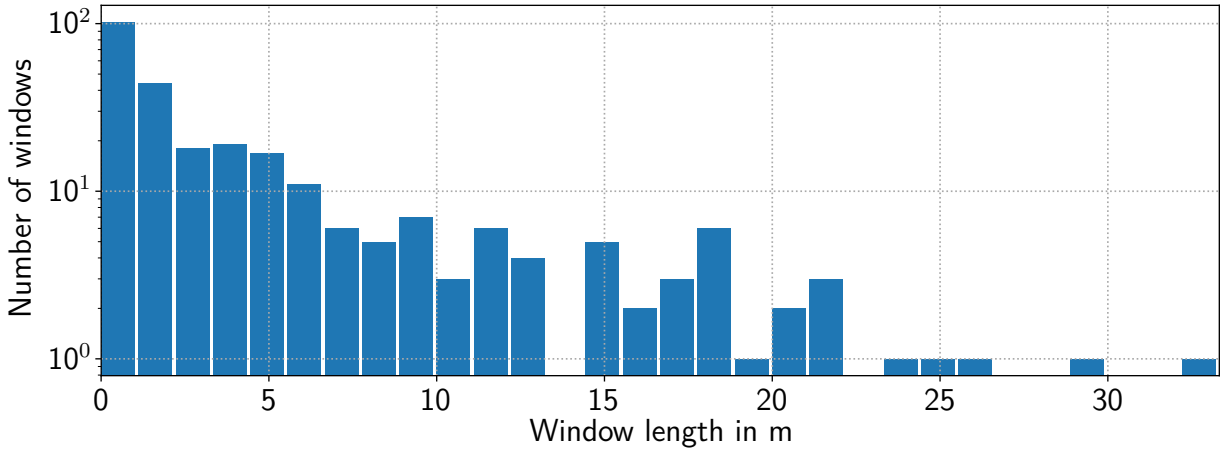


Figure 8.4: Histogram of window lengths in T_1. The number of windows with a given length is visualized. Note that the number of windows is logarithmically scaled.

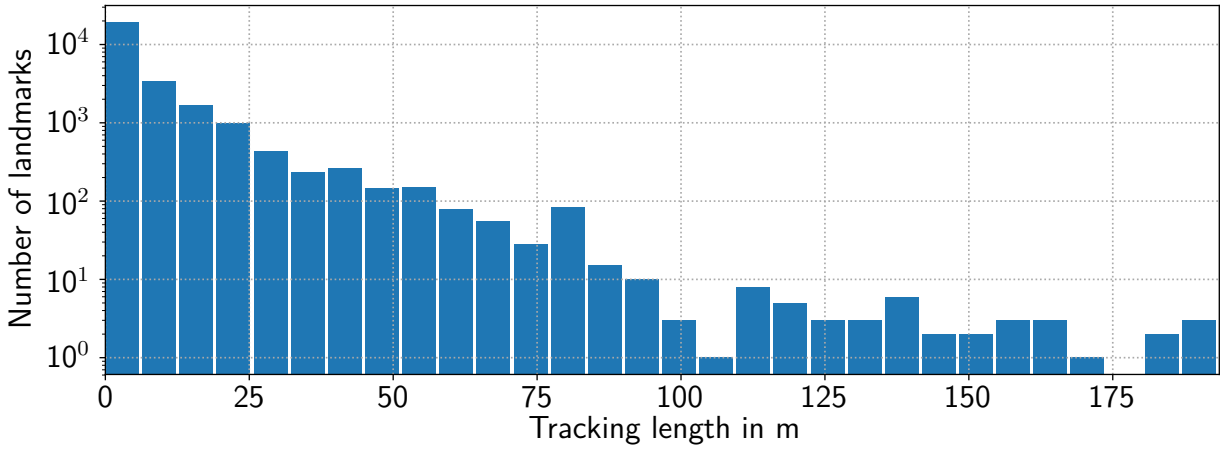


Figure 8.5: Histogram of tracking lengths of landmarks in T_1. The number of landmarks that are tracked at a given length along the trajectory is visualized. The number of landmarks is logarithmically scaled.

Since the start of a new window implies the insertion of a keyframe, we can maintain a conservative keyframe insertion policy that inserts as few as possible and as many as needed keyframes depending on how well features are tracked within the window. Interestingly, landmarks can exist beyond windows as implied in Figure 8.5: The tracking length of a large portion of the tracked landmarks is longer than the maximal window length of $33\,\mathrm{m}$. The longest track of landmarks measures almost $200\,\mathrm{m}$. Further investigation reveals that those features correspond to distant features that provide poor constraints on the relative translation but very restrictive constraints on the relative orientation for the odometry computation. In Table 8.2, we can see in the first row that while the average window length is $4.64\,\mathrm{m}$, the average tracking length of $6.91\,\mathrm{m}$ is larger, stressing that on average more landmarks are tracked beyond the window length.

However, this is not the case for all trajectories. For instance, in 0020, 0027, 0028, 0033, 0034^*, and 0034, the average tracking length is smaller than the average window length. This implies that the features are not as stably tracked as in the other datasets. The reason for the varying tracking performances can have different causes that we will analyze in the

Dataset	Average window length	Average number of detected landmarks	Average landmark tracking length
T_1	4.64 m	256	6.91 m
T_2	3.46 m	430	3.5 m
0018	3.89 m	204	5.06 m
0020	3.95 m	387	3.28 m
0027*	3.58 m	516	5.05 m
0027	3.1 m	658	2.91 m
0028	2.44 m	656	1.69 m
0033	2.9 m	278	1.7 m
0034*	2 m	98	1.38 m
0034	2.55 m	339	1.228 m
0071	2.88 m	192	3.42 m

Table 8.2: Front-end evaluation results for all trajectories. While the first column provides the average window length of a batch, the second column provides the average number of detected landmarks. Note that the average is only based on landmarks initially detected in a keyframe and successfully tracked in at least one subsequent frame. The third column provides information on the average tracking length of successfully tracked features.

following. Further investigation reveals that T_1 provides excellent conditions for the feature tracking: Many close texture-rich facades combined with high-resolution cameras with a large stereo baseline of 0.85 m provide robust visual features with low reconstruction uncertainty (cf. Table 8.3) with very low average reprojection errors. Suppose distinct and well-trackable features are detected in the images. In that case, our conservative keyframe insertion policy assures that only a few good features are tracked, as is the case for T_1 and 0018 according to Table 8.2. In contrast, in 0027 and 0028, the average number of detected and tracked landmarks is very high. The reason for this is that the extracted features are not stable. That is why those datasets' average tracking lengths are significantly smaller than T_1 and 0018. 0034^* has the lowest number of successfully detected and tracked features. Further investigation reveals that a significant part of the images capture bushes and trees on which many features are detected. However, due to the similarity, they are not tracked well. Consequently, the average tracking length of landmarks is comparatively small, as indicated in Table 8.2. Since not many features are tracked well, more keyframes that shorten the window lengths need to be inserted. In general, the feature tracking performs better in T_1 and T_2 as the cameras provide images with higher resolution than in the KITTI datasets.

In summary, the visual feature detection and tracking in the front-end heavily depends on the structure of the scene and the image resolution. Features are tracked well if the scene contains texture-rich parts close to the sensor system. The front-end provides comparatively large tracking windows in which the landmarks are tracked along a large trajectory, enabling stable back-end processing. If too much vegetation is captured by the images or too many texture-less parts are in the images, the feature tracking suffers.

8.1.2 Back-End Evaluation

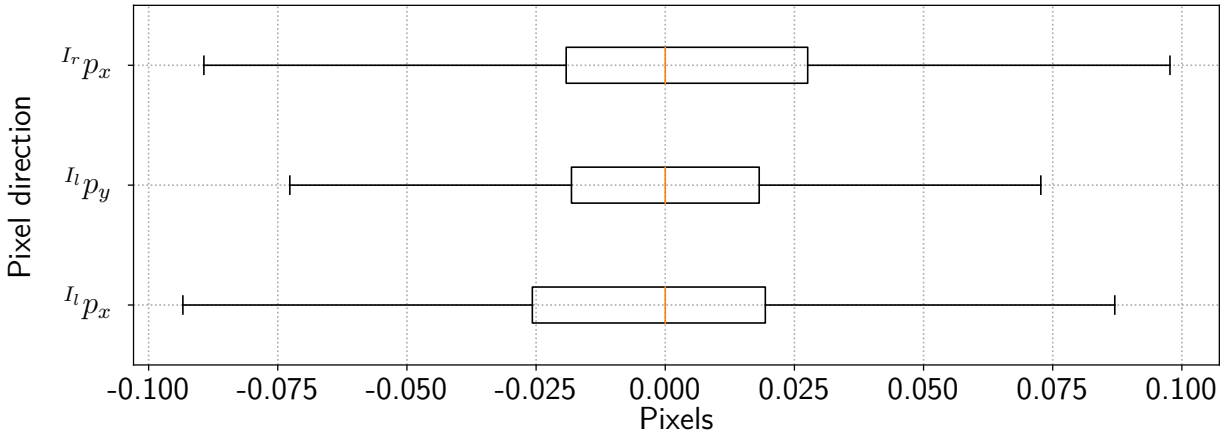
In the back-end, we perform the windowed bundle adjustment on each window the front-end provides. The windowed bundle adjustment has two primary purposes. First, since we apply a robust optimization approach, feature observations that lead to high residuals are identified as outliers. Second, the remaining residuals for the feature observations identified as compatible observations are used for the uncertainty estimation for the observations in the interval-based odometry computation as discussed in Subsection 4.3.2. Accordingly, we divide the evaluation into two parts. First, we provide experimental results on the windowed bundle adjustment in Part 8.1.2.1 evaluating the reprojection errors and the landmark reconstruction uncertainties. Second, we evaluate the interval-based odometry estimation in Part 8.1.2.2. We conclude the back-end evaluation by directly comparing the uncertainty estimates of our interval-based odometry approach and a classical MLE with a Gaussian uncertainty model applying covariance propagation for the uncertainty estimation of the relative transformation in Part 8.1.2.3.

8.1.2.1 Windowed Bundle Adjustment

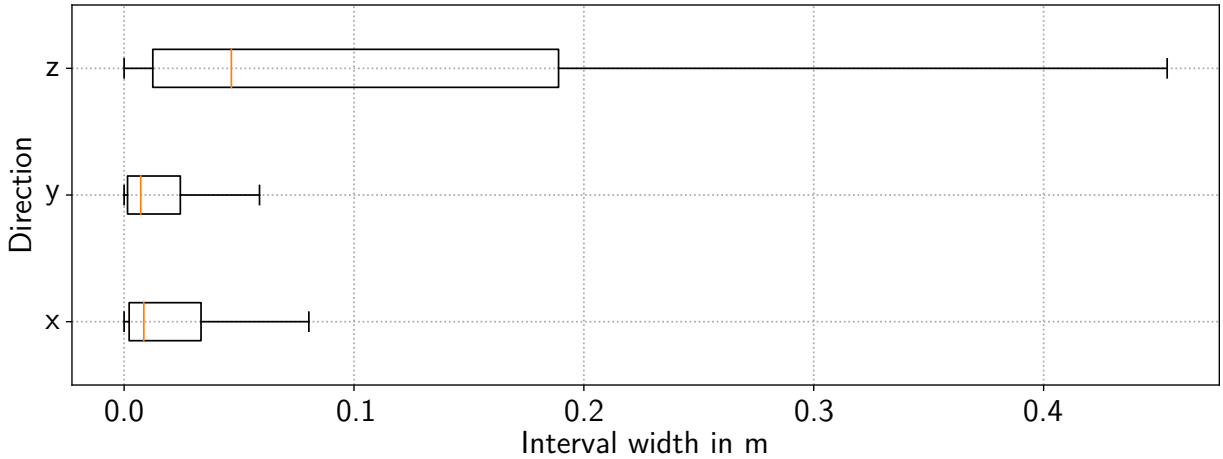
The main two objectives of the windowed bundle adjustment are to estimate the poses and the landmark locations so that the reprojection errors are minimized in the least squares sense and the outliers that exceed a maximal permitted reprojection error are identified and excluded from the minimization. In contrast to the classical approach, where individual feature observations are treated as outliers, we stage landmarks as outliers or inliers. As explained in Subsection 4.3.1, if the landmark is connected to at least one observation that exceeds the maximal permitted reprojection error threshold, we define the landmark and all connected observations as outliers. Since the threshold selection depends on the image resolution, we must select different thresholds for the author-collected and KITTI datasets. For the author-collected datasets T_1 and T_2, we choose a maximal reprojection error of $3.0\,\text{px}$, and for the KITTI datasets, we choose $4.0\,\text{px}$. The thresholds are selected empirically.

In Figure 8.6, the landmark-related uncertainties in T_1 are exemplarily visualized. While Figure 8.6a presents the reprojection errors of the inlier observations after the windowed bundle adjustment, Figure 8.6b shows the interval-based reconstruction uncertainty of the landmarks in the local frames C_l. Table 8.3 shows results for all datasets using further condensed evaluation metrics.

According to Figure 8.6a, most inlier observations have a reprojection error below $0.1\,\text{px}$ in T_1. As depicted in Table 8.3, the average absolute error measures $0.06\,\text{px}$, showing that the bundle adjustment achieves subpixel accuracy. The presented reprojection errors do not include outliers. As shown in the first column, in T_1 among all detected landmarks $10\,\%$ are identified as outliers along the whole trajectory. In Figure 8.7, we qualitatively illustrate inlier and outlier landmarks for the first window processed in T_1 for the same scene we introduced exemplarily in Figure 4.15. Outliers are colored red, and inliers are green. The bundle adjustment correctly identifies all features on the moving vehicle as outliers since the reprojection error of features on moving objects results in observations with large residuals in the optimization. Furthermore, many outliers are detected on a construction barrier on the bottom right part. Here, the outliers mainly result from the incorrect association of features since similar features are detected on



(a) Reprojection errors after the windowed bundle adjustment for inlier observations. The reprojection errors are visualized as boxplots for the right image along the horizontal image axis and the left image along the vertical and horizontal image axes.



(b) Uncertainties of reconstructed landmarks. The boxplots visualize the interval widths of the reconstructed 3D features in the left camera frame C_l.

Figure 8.6: Landmark reconstruction uncertainties. Figure 8.6b shows the reconstruction uncertainty and Figure 8.6a the reprojection errors in T_1. The reprojection errors are used as the pixel uncertainties for stereo-camera-based landmark reconstruction.

the repetitive texture that is incorrectly associated during the tracking in the window. As a result, the windowed bundle adjustment performs a good preselection of consistent landmarks, which can be used for the interval-based odometry computation.

Note that the average absolute reprojection error for all KITTI trajectories is larger since we admit larger residuals for inlier observations. The boxplots in Figure 8.6a reveal that the median of the reprojection error is very close to zero. The reprojection error has approximately a Gaussian distribution with zero means. This indicates that the least squares result represents the most consistent result considering the observations. This underlines that the approximation of the uncertainty of the pixel observation with the reprojection error is indeed tenable.

The 3D uncertainty of the reconstructed landmarks as interval boxes using the reprojection errors is shown in Figure 8.6b. The interval box is computed with the stereo feature contractor and depth refinement with LiDAR measurements as presented in Subsection 4.2.2. Note that in C_l, the z-axis points to the driving direction, x to the right, and y downward. The

Figure 8.7: Outliers are detected and deleted by the robust windowed bundle adjustment. Successfully tracked features in the window that are detected as invalid due to high residuals in the windowed bundle adjustment are colored red, and good features that are permitted for the interval-based odometry computation are colored green.

uncertainty in the depth axis (z) is the largest due to the stereo camera parallax, although we incorporate LiDAR data. Only a small part of the landmarks are depth refined by the LiDAR. This is because the LiDAR measurements are less dense than the camera data. Notably, the uncertainty in the y-axis is smaller compared to the other directions. The explanation for this observation is the admitted error in the vertical direction when the left and right image features are matched: Although we stereo-rectify the image pairs for each frame, the rectification as well as the feature detection algorithm are not perfect due to which corresponding features are not always detected precisely on the same image rows. That is why we tackle this matching problem by admitting feature matching across at a maximum of 2.0 px above or below the query row in the right image. However, this also means that the matched features generate cones that have a smaller intersection region in the vertical direction, which coincides with the y-axis. Consequently, the uncertainty estimate in the y-axis is smaller as the intersection region is more susceptible to stereo-matching uncertainties.

According to Table 8.3 in T_1 and T_2 the median uncertainty in the x and y direction is below 0.01 m, while in z the uncertainty is 0.047 m and 0.022 m, respectively. However, for the KITTI datasets, the uncertainties are significantly larger. The main reason is that the reprojection error is twice as large as for the author-collected datasets. As explained above, this is due to the smaller resolution of the used cameras. Furthermore, in the KITTI datasets, the Velodyne HDL64E only captures a small part of the images, so the depth uncertainty is on average larger in the KITTI trajectories. In 0020, the uncertainties are huge because, in this

Dataset	Ratio of invalid landmarks	Average absolute reprojection error	Median landmark uncertainty in x	Median landmark uncertainty in y	Median landmark uncertainty in z
T_1	$10\,\%$	0.06 px	0.009 m	0.007 m	0.047 m
T_2	$4\,\%$	0.05 px	0.005 m	0.002 m	0.022 m
0018	$13\,\%$	0.13 px	0.082 m	0.026 m	0.277 m
0020	$8\,\%$	0.18 px	0.344 m	0.085 m	1.02 m
0027*	$9\,\%$	0.13 px	0.064 m	0.023 m	0.223 m
0027	$4\,\%$	0.16 px	0.103 m	0.037 m	0.351 m
0028	$5\,\%$	0.12 px	0.064 m	0.02 m	0.21 m
0033	$4\,\%$	0.13 px	0.204 m	0.074 m	0.619 m
0034*	$2\,\%$	0.12 px	0.104 m	0.04 m	0.364 m
0034	$3\,\%$	0.12 px	0.183 m	0.074 m	0.572 m
0071	$14\,\%$	0.1 px	0.035 m	0.013 m	0.1 m

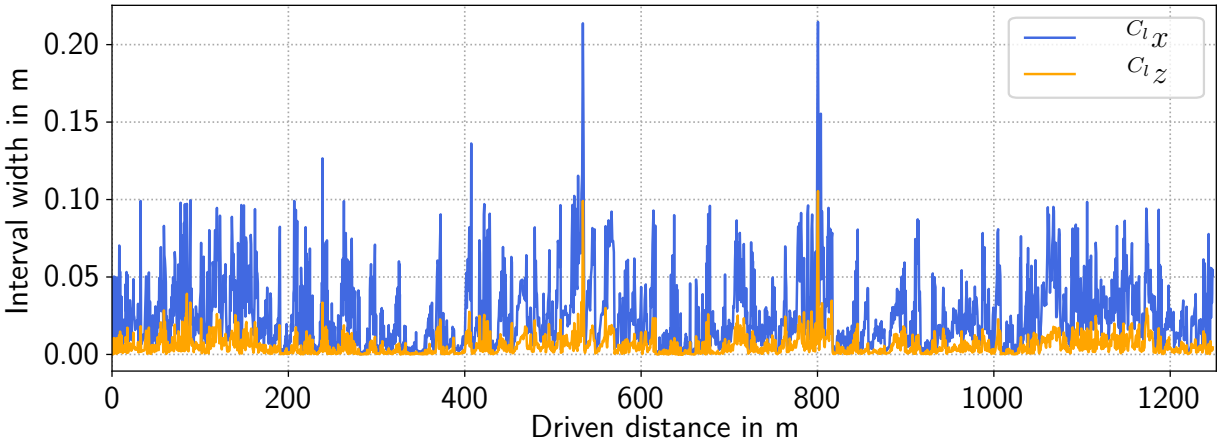
Table 8.3: Landmark-related evaluation of the windowed bundle adjustment for all trajectories. The first column depicts the ratio of landmarks staged as outliers in the bundle adjustment since they are connected to observations that exceed the maximal permitted reprojection error (residual). The second column lists the average reprojection errors of feature observations corresponding to non-outlier landmarks. The last three columns show the median landmark uncertainties measured by the interval width of the reconstructed landmark using the proposed interval-based approach described in C_l.

trajectory, the camera captures mainly features that are far away, as only a few close objects are captured by the sensors. Hence, only very few good features are tracked, and the landmark uncertainty suffers.

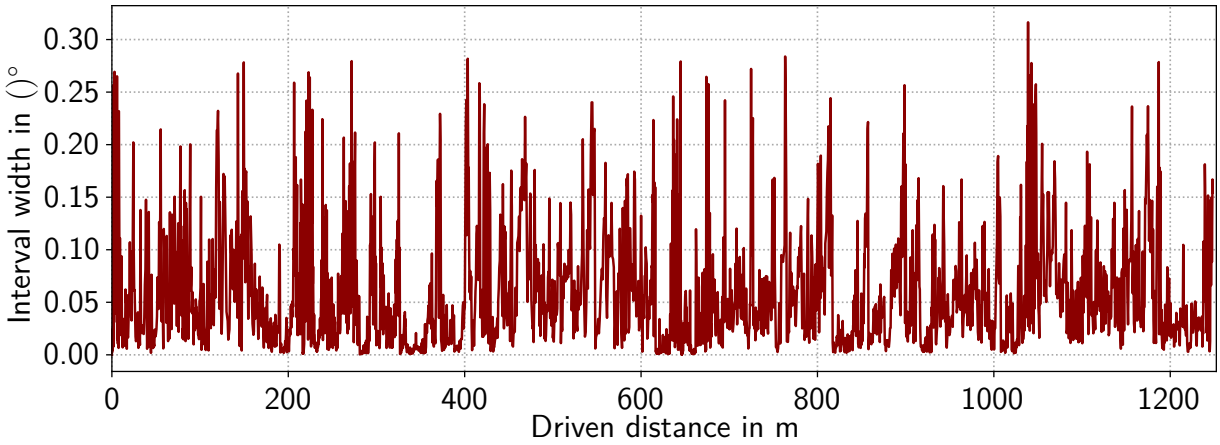
8.1.2.2 Interval-based Odometry

Now we focus on the interval-based odometry that poses the output of the visual odometry module and is further used in the localization modules in the HyPaSCoRe Localization pipeline. Figure 8.8 depicts the interval widths of the frame-to-frame relative transformations along the trajectory T_1. We only consider the two translation components x and z in C_l in the driving plane and the yaw-rotation angle. We omit the evaluation of the other components since they are not relevant for the HyPaSCoRe Localization, as only the 3 DOF relative transformation is passed to the localization modules. In Table 8.4, we summarize the results for all trajectories. While the rotation interval width visualized in Figure 8.8b varies along the trajectory, it does not exceed $0.35°$. According to Table 8.4, the median interval width is $0.04°$. Hence, the rotation uncertainty is contracted to very small values.

We can see a similar behavior of the interval widths for the translation components in Figure 8.8a. Although the x-component has two peaks to an interval width of approximately $20\,$cm, the z component stays under $5\,$cm. It is noticeable that the uncertainty of the x-component is always larger than the z-component. However, this observation seems to be contradicting to the results discussed in the previous section: According to Table 8.3, the



(a) Frame-to-frame translation uncertainty. We only visualize the uncertainties on the ground plane described in C_l, omitting the elevation axis.



(b) Frame-to-frame rotation uncertainties. We only visualize the yaw angle in the C_l.

Figure 8.8: Uncertainty evaluation of the frame-to-frame interval-based odometry in T_1 along the trajectory.

z-uncertainty of the reconstructed landmarks is always larger than in the other directions, and therefore we would expect a larger uncertainty in the z-direction of the translation. Further investigations reveal that the intersection of all contractors $\mathcal{C}_i^{\text{SVO}}$ leads to the smaller uncertainty of the z-translation component. While applying a single Stereo-Visual-Contractor $\mathcal{C}_i^{\text{SVO}}$ just for the i-th landmark, the uncertainty along the driving direction z is – as expected – the highest compared to the other directions. However, if we intersect all contractors $\mathcal{C}_i^{\text{SVO}}$ to $\mathcal{C}_{\text{SVO}}$ according to (4.15), the uncertainty in z is reduced as the depth estimation of the landmarks are more susceptible to the image measurement perturbations.

According to Table 8.4, the odometry estimates for the KITTI datasets are significantly more uncertain than the author-collected datasets. As described above, the higher reprojection errors are due to lower image resolution. In particular, 0020 has a comparatively large uncertainty for the estimated odometry since too few features are detected and tracked. Comparing the different datasets, the first column in Table 8.4 reveals that the vehicle was operated in the KITTI trajectories with higher speeds. Hence, another reason for the lower tracking performance in the KITTI datasets may be motion blur caused to higher velocities.

Dataset	Average frame-to-frame distance	Median interval width ${}^{C_l}x$	Median interval width ${}^{C_l}z$	Median interval width rotation
T_1	$0.429\,\text{m}$	$0.017\,\text{m}$	$0.003\,\text{m}$	$0.04°$
T_2	$0.38\,\text{m}$	$0.004\,\text{m}$	$0.001\,\text{m}$	$0.012°$
0018	$0.819\,\text{m}$	$0.046\,\text{m}$	$0.009\,\text{m}$	$0.091°$
0020	$1.16\,\text{m}$	$0.072\,\text{m}$	$0.015\,\text{m}$	$0.126°$
0027*	$0.667\,\text{m}$	$0.039\,\text{m}$	$0.007\,\text{m}$	$0.088°$
0027	$0.817\,\text{m}$	$0.045\,\text{m}$	$0.008\,\text{m}$	$0.099°$
0028	$0.81\,\text{m}$	$0.045\,\text{m}$	$0.008\,\text{m}$	$0.099°$
0033	$1.088\,\text{m}$	$0.06\,\text{m}$	$0.01\,\text{m}$	$0.108°$
0034*	$0.77\,\text{m}$	$0.04\,\text{m}$	$0.007\,\text{m}$	$0.096°$
0034	$1.106\,\text{m}$	$0.061\,\text{m}$	$0.012\,\text{m}$	$0.124°$
0071	$0.227\,\text{m}$	$0.021\,\text{m}$	$0.004\,\text{m}$	$0.053°$

Table 8.4: Interval-based odometry evaluation. In this table, the frame-to-frame odometry computed with $\mathcal{C}_{\text{SVO}}$ is presented. The first column shows the average frame-to-frame distance between two consecutive poses. This value provides an average value of how fast the vehicle was moved. The second and third columns depict the median interval widths for the translation components on the driving plane. The fourth column provides the interval width for the yaw rotation angle. All relative transformation parameters are described in the local left camera frame C_l.

8.1.2.3 Interval-based vs. MLE-based Odometry

To compare our interval-based odometry with a State of the Art approach, we use a small part of T_1. As a baseline approach in the literature, we use an MLE approach with a Gaussian uncertainty model that implements the least squares optimization in a graph optimization fashion. This enables the direct comparison since we can reuse the SLAM-graph for the baseline approach. As the graph optimization library, we use g^{2}o [29] that we also use for the windowed bundle adjustment in our approach. The uncertainty is determined by the system covariance matrix obtained by inverting the systems matrix as explained in Subsection 2.1.3. We retrieve the pose covariance matrices by cutting the respective blocks out of the whole system covariance matrix. We compare the $99.9\,\%$-confidence ellipsoids with our interval estimates.

In Figure 8.9 the results are shown. The pose uncertainty estimated by the interval method rises continuously as the size of the pose boxes increases. Also, the pose uncertainty of the MLE results grows as the size of the $99.9\,\%$-position-ellipsoids increases, as shown in the close-ups. However, the uncertainty rises much slower than the interval uncertainty: While the semi diagonal of the last position box computed by the interval-based odometry measures $6.55\,\text{m}$ for a driven distance of $74.8\,\text{m}$, the semi diagonal of the corresponding $99.9\,\%$-position-ellipsoid computed by g^{2}o is only $0.44\,\text{m}$. As a result, our method accounts for a drift of $8.7\,\%$, while the MLE method estimates a drift of $0.58\,\%$. That means g^{2}o's uncertainty estimate is nearly 15 times more optimistic than the interval method.

However, let us inspect the deviation of the MLE results from the ground truth. We can see that the estimate is overconfident: As shown in the second close-up compared to the first, the

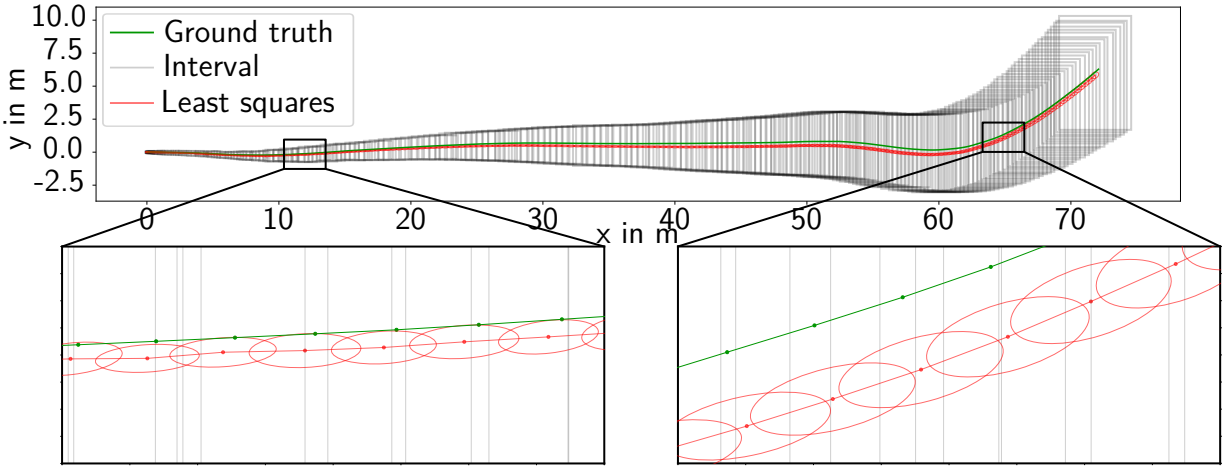


Figure 8.9: Comparison of the interval-based odometry with a least squares approach (MLE with Gaussian uncertainty) for a small part of T_1. The gray boxes visualize the interval results while the red ellipses represent the $99.9\,\%$-confidence ellipsoids computed by the g^2o graph optimization library. The ground truth trajectory is colored green. Both close-up figures have the same scale.

deviation between ground truth and g^{2}o pose estimates also rises, and the $99.9\,\%$-ellipsoids do not contain ground truth anymore. Considering the deviation between ground truth and g^{2}o position estimate, we have an actual drift of $0.72\,\%$, which is higher than the estimated drift of $0.58\,\%$. In other words, the MLE optimization method underestimates the uncertainty of the poses with the $99.9\,\%$-ellipsoids.

Our proposed method shows a contrary behavior: It estimates a very high drift of $8.7\,\%$ and is very pessimistic. The estimated bounds for the vehicle pose always enclose ground truth relative motion estimate. Hence, this behavior is beneficial for safety-critical systems.

8.1.3 Runtime

Finally, we want to present the runtime evaluation of the visual odometry module. As presented in Section 7.1, the visual odometry consists of two threads – one for the front-end and the second for the back-end. The front-end processes the incoming sensor data, representing the real-time critical process in the visual odometry module. In contrast, the back-end simultaneously performs the windowed bundle adjustment and the interval-based odometry computation for the whole window. Since a window consists of multiple frames and the sensors are operated at $10\,\text{Hz}$, the back-end typically has an operation time frame of several seconds. Since the windowed bundle adjustment with g^{2}o and the interval-based odometry computation for each frame in the window are typically processed in less than a second, the back-end is not real-time critical. That is why we will only focus on the runtimes in the front-end.

Figure 8.10 shows the overall runtimes for the front-end for the trajectory T_1. A dashed red line illustrates the real-time runtime limit. We can see that the runtimes are below the limit except at the beginning, where sudden runtime peaks exceed the limit. Note that we are not using any real-time kernels so the runtime peaks may be caused by other parallel processes called by the operating system.

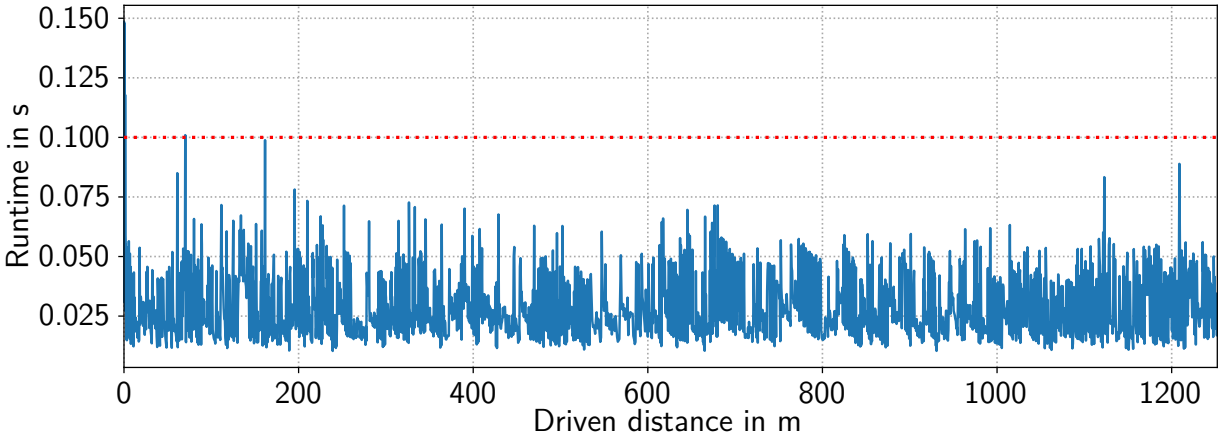


Figure 8.10: Runtimes of the real-time critical front-end along the trajectory of T_1.

Dataset	Front-end	Preprocessing	Tracking	Tracking and detection	Frame-to-frame pose optimization
T_1	0.029 s	0.006 s	0.003 s	0.034 s	0.018 s
T_2	0.038 s	0.006 s	0.005 s	0.038 s	0.024 s
0018	0.034 s	0.002 s	0.004 s	0.025 s	0.023 s
0020	0.033 s	0.003 s	0.003 s	0.026 s	0.021 s
0027*	0.036 s	0.002 s	0.004 s	0.021 s	0.026 s
0027	0.032 s	0.002 s	0.003 s	0.024 s	0.021 s
0028	0.032 s	0.003 s	0.003 s	0.026 s	0.02 s
0033	0.037 s	0.003 s	0.003 s	0.028 s	0.022 s
0034*	0.033 s	0.002 s	0.003 s	0.025 s	0.019 s
0034	0.037 s	0.002 s	0.004 s	0.028 s	0.02 s
0071	0.034 s	0.003 s	0.005 s	0.024 s	0.025 s

Table 8.5: Runtimes in the front-end. The runtimes per frame for the whole front-end and the different subprocesses are listed for all datasets. All processing times are averaged along the whole trajectories.

In Table 8.5, the average processing times for the entire front-end and the different subprocesses are listed. For all datasets, the average front-end processing time is significantly lower than the time limit for real-time execution. Note that the tracking is very lightweight, while the tracking combined with the detection takes more time. Fortunately, only when a new keyframe is inserted the detection is triggered. The insertion of a keyframe happens rarely and therefore saves computation time. Furthermore, the preprocessing of the images, including stereo-rectification and undistortion, takes almost twice the time for the author-collected datasets T_1 and T_2 compared to the KITTI datasets as the image resolution in the author-collected datasets is higher. Also, the tracking and detection times are longer for T_1 and T_2. Nonetheless, we can conclude that the visual odometry module is generally lightweight enough to achieve real-time performance.

8.2 Coarse Localization

Now we focus on the coarse localization. In the first part of the experimental evaluation, we will consider the interval-based localization in Subsection 8.2.1. In the second part, we compare the coarse localization with a State of the Art AMCL approach in Subsection 8.2.2. Finally, we conclude the evaluation of the coarse localization by a runtime analysis.

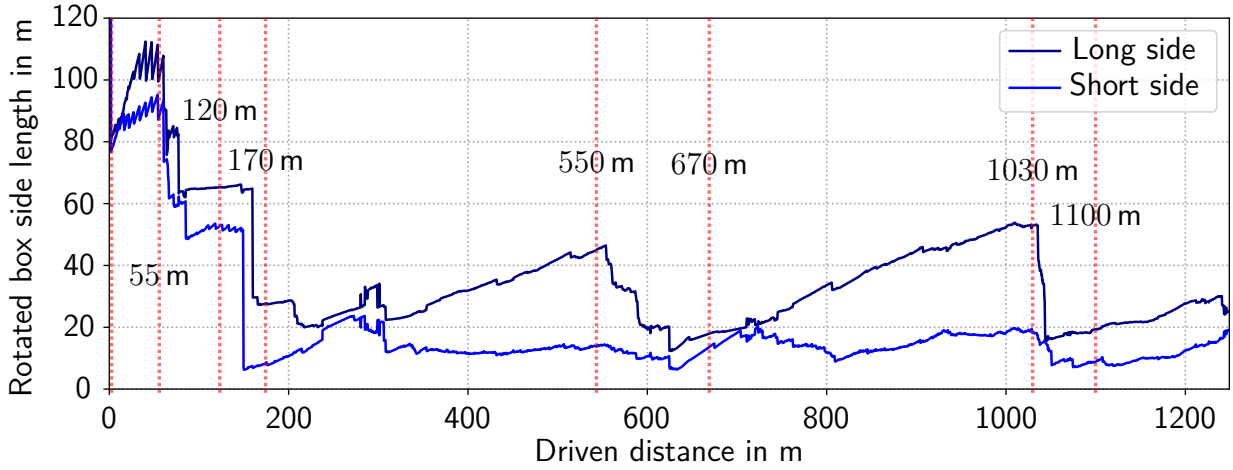
8.2.1 Interval-based Localization

We pick two exemplary trajectories – T_1 from the author-collected and 0027 from the KITTI dataset – to evaluate the interval-based localization approach. Note that the interval-based approach provides regions. Therefore, standard evaluation metrics like the root mean squared error are not applicable. Hence, first, we will introduce the evaluation metrics and then examine the experimental results for the two representative trajectories.

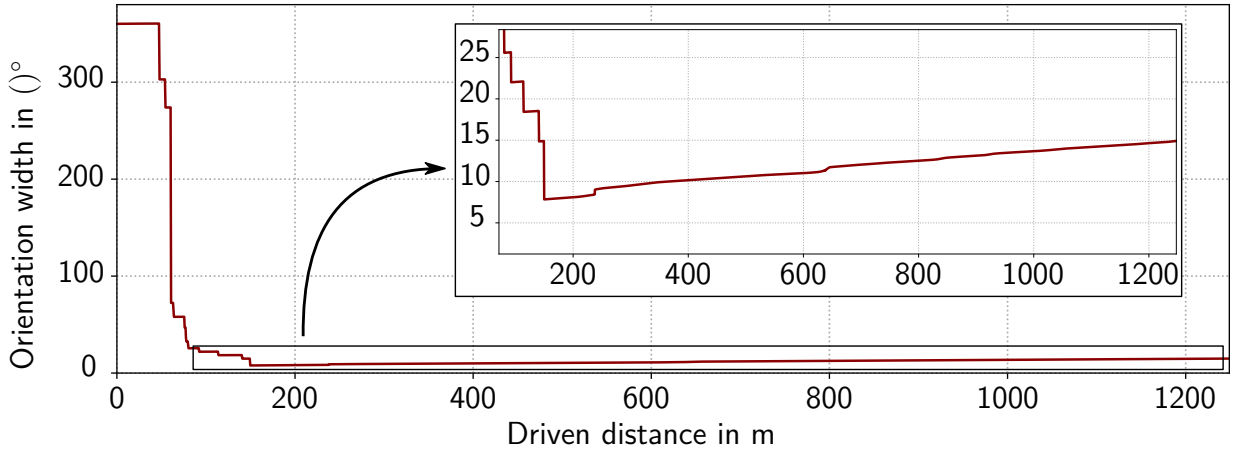
As the goal of the interval-based localization is to enclose the feasible set of poses as closely as possible, the quality of the interval method is well reflected by the sizes of the enclosing sets. As the 2D pose contains a translation and rotation part, we must examine the parameter sets independently. While the translation is represented by a subpaving composed of multiple 2D boxes, the rotation is represented by a 1D subpaving. Note, that to each rotation subset a translation subpaving is associated. As subpavings may contain disconnected subsets, evaluating the sizes of the enclosing sets is not as straightforward as if we would have connected sets. In the scope of this work, we evaluate the translation estimate by wrapping the smallest rotated box around the translation subpaving. By doing this, we obtain a convenient and intuitive way to describe the uncertainty by only considering the side lengths of the rotated boxes. Similarly, we define an enclosing interval hull around the feasible set of orientations – thus, also providing a simple metric on the size of the subpaving. However, the hull-based metric does not consider the size of infeasible gaps inside the enclosure. Figure 8.12c visualizes the feasible translation subpaving in green and the smallest rotated enclosing box in blue. Note that the feasible set consists of many disconnected sets. Although the total area of the green region is comparatively small, the box hull measures a comparatively large region. Nonetheless, this metric provides a convenient and simple geometric measure of the feasible set, so we decided to utilize it in our evaluation.

8.2.1.1 Author-Collected Dataset T_1

First, let us examine the author-collected dataset T_1. In Figure 8.11a, we visualize the rotated box hull side lengths along the driven distance. We evaluate the interval-based position estimate by considering the long and short sides of the rotated box. By decoupling the side lengths into two perpendicular directions represented by the side lengths, we can detect if the uncertainty grows isotropically or if there is a favored direction in which the uncertainty grows faster. The red dashed lines for selected frames indicate the driven distance for which a snapshot to the localization estimate is displayed in Figure 8.12. Figure 8.11b shows the interval hull of the orientation subpaving.



(a) Side lengths of the rotated box that encloses the feasible set of positions at each frame.

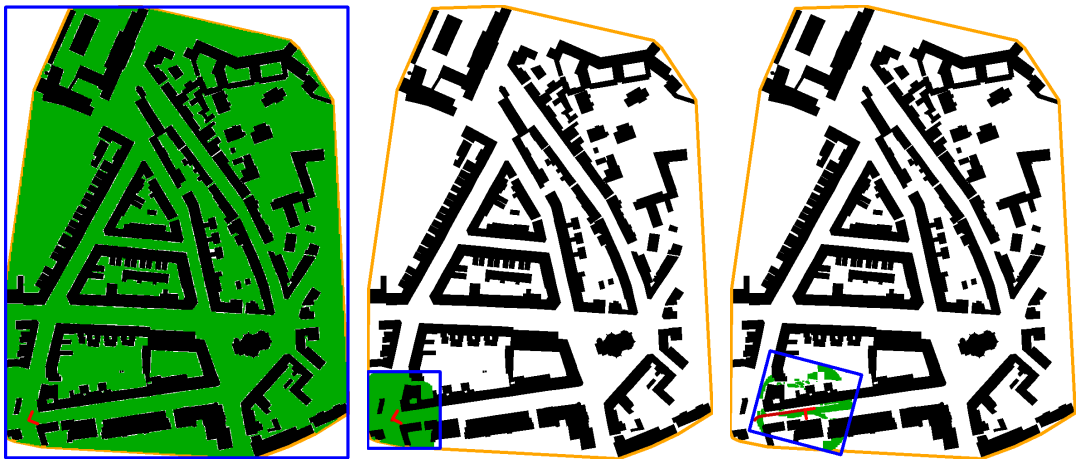


(b) Width of the orientation interval that encloses the feasible set of orientations.

Figure 8.11: Sizes of the enclosing intervals for the position (Figure 8.11a) and the orientation (Figure 8.11b) for trajectory T_1. The sizes represent the uncertainty of the interval-based pose estimation. The according feasible set of positions are visualized for selected frames in Figure 8.12. The frames are highlighted by red dashed lines in Figure 8.11a.

Initially, as illustrated in Figure 8.12a, the vehicle can potentially be located everywhere on the map. Accordingly, the position uncertainty is very large. Due to scaling reasons, in Figure 8.11a, we limit the plot to $120\,\text{m}$ of uncertainty. However, shortly after initialization, GNSS measurements are available. Using the GNSS Contractor, the feasible set is reduced to the circular uncertainty region of the GNSS measurement as shown in Figure 8.12b. Since we fix the GNSS uncertainty to a radius of $50\,\text{m}$, the size of the boxes shrinks to approximately $80\,\text{m}$. We do not reach the expected $100\,\text{m}$ side length, as the vehicle is closely located on the map's border. As the No-Overlap Contractor limits the size of the feasible region so that it stays inside the map, the feasible region is accordingly smaller.

In the first $55\,\text{m}$, we can observe a serrated pattern of the box side lengths in Figure 8.11a. This pattern is caused by GNSS measurements that restrict the feasible set to the GNSS-uncertainty region. When the GNSS Contractor performs the contraction, the feasible set is contracted, while uncertain odometry updates let the feasible region continuously grow. That is also why the long side is consistently contracted to $100\,\text{m}$, which corresponds to the diameter



(a) Driven distance: 0 m. (b) Driven distance: 2 m. (c) Driven distance: 55 m.

(d) Driven distance: 120 m. (e) Driven distance: 170 m. (f) Driven distance: 550 m.

(g) Driven distance: 670 m. (h) Driven distance: 1030 m. (i) Driven distance: 1100 m.

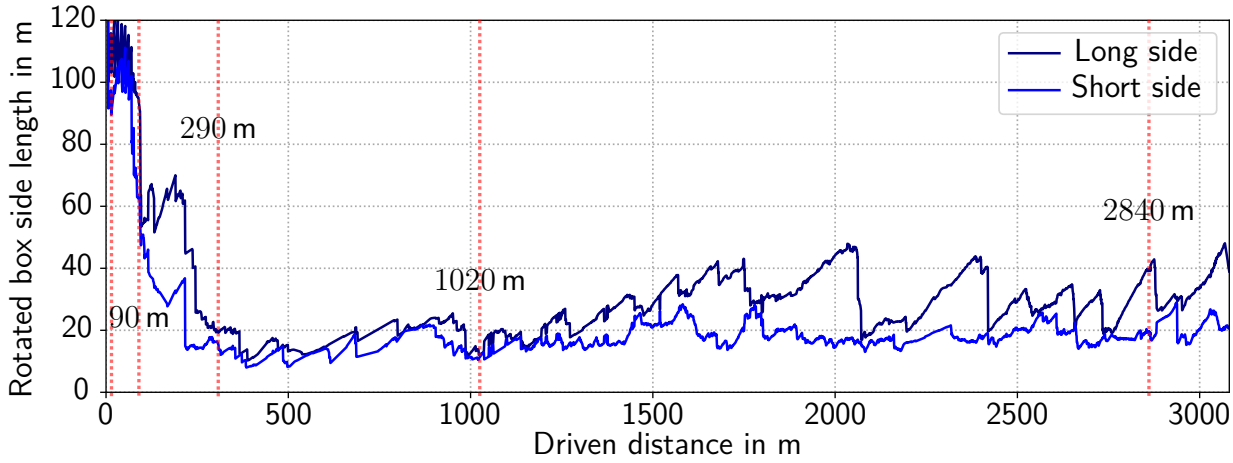
Figure 8.12: Sequence of localization results of the interval method at different time steps for trajectory T_1. All feasible poses of all rotation bins are combined into each figure. The blue box is the smallest rotated box enclosing the feasible positions set. Figure 8.11a displays the sizes of the blue box.

of the GNSS uncertainty region. While at $55\,\mathrm{m}$ (Figure 8.12c) the feasible set already has many disconnected subsets, at $120\,\mathrm{m}$ (Figure 8.12d) only two comparatively small disconnected subsets remain. At around $120\,\mathrm{m}$, we again can observe the same serrated pattern of the short side of the rotated box hull due to the feasible subset close to the border region of the GNSS-uncertainty region (see Figure 8.12d).

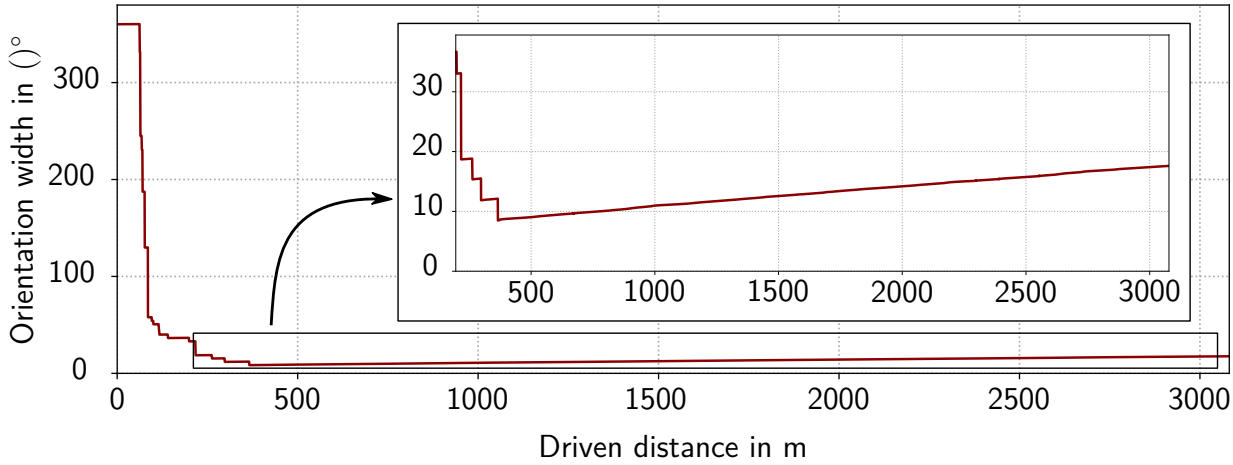
After the driven distance of $160\,\mathrm{m}$, the bottom subset shown in Figure 8.12d is entirely deleted from the feasible set. That is why the rotated box hull size significantly drops at $160\,\mathrm{m}$. The remaining subpaving is visualized in Figure 8.12e. Note the elongated shape of the feasible set along the driving direction. This is due to the building constellation that only makes contractions in the lateral direction possible, as buildings are only located on the left and right sides of the vehicle. That is why the short side length has a comparatively stable length of approximately $10\,\mathrm{m}$ while the long side length increases continuously between $160\,\mathrm{m}$ and $550\,\mathrm{m}$ driven distance. Figure 8.12f reveals that in this part of the trajectory, we only have a straight road with buildings on the left and right sides. Hence, the uncertainty grows along the driving direction. As a result, at the end of this trajectory part, the uncertainty in the driving direction is comparatively large and reaches almost $45\,\mathrm{m}$ according to Figure 8.11a. However, at a driven distance of $560\,\mathrm{m}$, the No-Cross Contractor makes the contraction along the driving direction possible since a perpendicular wall is perceived. Accordingly, the uncertainty can be reduced again between $560\,\mathrm{m}$ and $650\,\mathrm{m}$ driven distance as depicted in Figure 8.11a. Figure 8.12g shows the remaining feasible set of positions.

When the vehicle turns and moves in another direction also leads to a significant contraction of the feasible set. After the feasible set of translations is narrowed to a comparatively small set until a driven distance of $160\,\mathrm{m}$, there are two significant turns at $600\,\mathrm{m}$ and $1040\,\mathrm{m}$ the vehicle performs. If we consider the rotated box hull side lengths in Figure 8.11a, the contractions directly correspond to those turns. For instance, consider the 90° turn at $1040\,\mathrm{m}$. Before the turn, the position uncertainty is prominent along the driving direction as shown in Figure 8.12h. The larger side of the rotated box has a length of $53\,\mathrm{m}$. During the turn, those parts in the front and back of the vehicle are successively dismissed from the feasible set using the No-Overlap Contractor as those parts are shifted into the buildings. The result after the turn is shown in Figure 8.12i, and the largest side length of the rotated box hull drops to less than $20\,\mathrm{m}$. Turns reduce the uncertainty due to two facts: First, when it comes to a turn, in urban environments, often perpendicular building facades are observed that are used for contraction. Second, the feasible set of positions is shifted to another direction, so the buildings are observed from another direction. Further contraction based on the No-Cross Contractor and the No-Overlap Contractor is possible.

The rotation uncertainty shows a different behavior compared to the translation. While in the first $50\,\mathrm{m}$, we can observe a serrated pattern of the box side lengths in Figure 8.11a, the rotation uncertainty Figure 8.11b does not change at all. This behavior is because the rotation is only passively contracted using the translation sets. None of the introduced contractors performs a contraction on the rotation directly. However, as we divide the initial feasible set of orientations into multiple bins and apply the contractors successively to dismiss infeasible translation subsets from each bin ${}^{M}\mathcal{P}_{L,i}$, we passively contract the orientation if the translation set ${}^{M}\mathcal{T}_{L,i}$ becomes empty for a particular orientation bin. Hence, we cannot directly dismiss the whole set of translations for a bin initially. As a result, the orientation cannot be contracted.



(a) Side lengths of the rotated box that encloses the feasible set of positions at each frame.



(b) Width of the orientation interval that encloses the feasible set of orientations.

Figure 8.13: Sizes of the enclosing intervals for the position (Figure 8.13a) and the orientation (Figure 8.13b) for the KITTI trajectory 0027. The sizes represent the uncertainty of the interval-based pose estimation. The according feasible set of positions is visualized for selected frames in Figure 8.14. The frames are highlighted by red dashed lines in Figure 8.13a.

However, the rotation uncertainty drops quickly after a driven distance of $50\,\mathrm{m}$ as shown in Figure 8.11b. This is mainly caused by the GNSS Contractor: As the GNSS uncertainty region is fixed to $50\,\mathrm{m}$ radius. Those translation sets of bins oriented in the opposite direction of the actual orientation get deleted as those sets cannot be located inside the uncertainty region of the GNSS measurement. As a result, the orientation uncertainty rapidly decreases to $8°$ interval width at $170\,\mathrm{m}$ driven distance. Note that in our implementation, we perform continuous splitting of the rotation interval if a rotation bin's uncertainty exceeds a given threshold. This makes further contraction of the orientation uncertainty possible. However, in this example, the rotation splitting does not decrease the rotation uncertainty further. That is why the orientation uncertainty linearly increases with $0.00625° \frac{1}{m}$ due to the uncertainty accumulation of the odometry data after $170\,\mathrm{m}$.

8.2.1.2 KITTI Dataset 0027

Now we examine the KITTI dataset 0027. The rotated box hull side lengths and the orientation interval width along the trajectory are depicted in Figure 8.13a and Figure 8.13b, respectively. Figure 8.14 shows the snapshots to selected frames again highlighted by dashed red lines in Figure 8.13a.

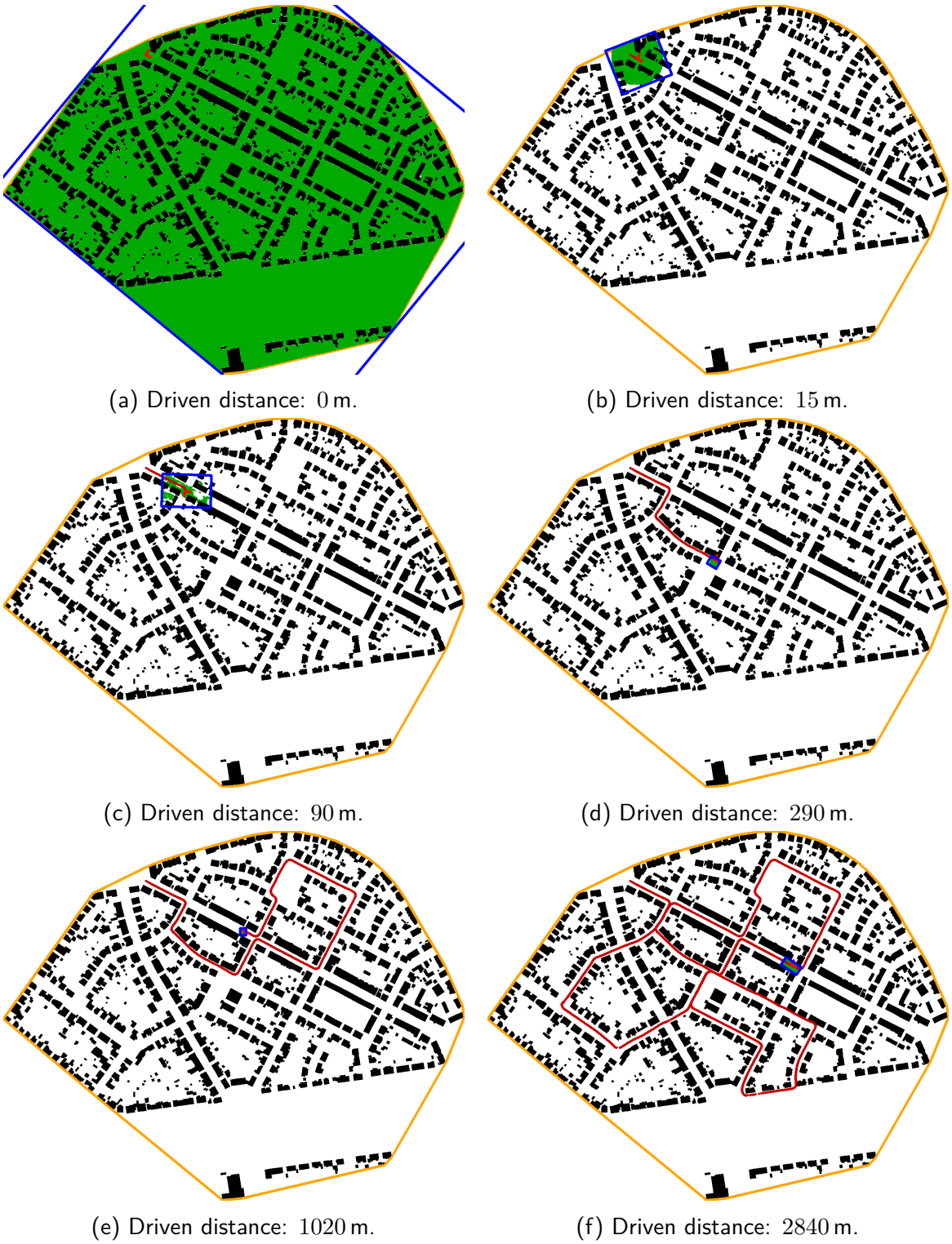
In this dataset, we can observe a similar behavior of our interval-based method as in T_1. The serrated pattern of the rotated box hull side lengths is also observable in the beginning. However, the size of the rotated box hull shrinks faster as the map and the trajectory are topologically different: In T_1, there are many connected buildings along the streets. However, in the considered part of Karlsruhe, the buildings are disconnected and are more distributed within the map. The more buildings are distributed, the more regions exist where the No-Overlap Contractor can dismiss infeasible translation subsets that are shifted into the buildings. Furthermore, as the buildings are not connected, more building facades perpendicular to the driving direction are visible, which improves the No-Cross Contractor performance. Another essential factor is also the trajectory. While in T_1, the first turn is at $190\,\text{m}$, in 0027, the first turn is already at $90\,\text{m}$. As explained above, turns provide valuable information for the contractors so that the feasible set is significantly reduced in 0027 earlier compared to T_1.

The GNSS information only provides helpful information at the beginning when the uncertainty is large. However, if the translation uncertainty is comparatively small, the GNSS contractor does not contribute to reducing the feasible set. Nonetheless, the GNSS Contractor is very important for our approach as it initially reduces the feasible set significantly.

The shape of the graph in Figure 8.13b is very similar to Figure 8.11b. As observed above, for 0027 the rotation is only contracted after a driven distance of $50\,\text{m}$ as shown in Figure 8.13b.

8.2.2 Hybrid Method vs. AMCL

In the last section, we analyzed the interval-based localization results. Now we will consider the bounded MCL part by comparing our method with a State of the Art Adaptive Monte Carlo Localization (AMCL). The AMCL combines an adaptive resampling approach [147] with a low variance resampling [84] similar to the approaches presented in [89, 91, 93, 94]. We initialize the AMCL with 20000 uniformly distributed particles similar to our hybrid method that initializes the location to the map region. We divide the initial rotation uncertainty into 25 bins for our hybrid method. As the bounded MCL allocates 20 particles per bin, our method only uses a maximum of 500 particles. For a fair comparison, the observation model for weighting the particles is the beam-end model presented in the bounded MCL approach described in Section 5.3. Note that the observation model is interchangeable with more sophisticated observation models [93, 94]. We selected the well-known beam-end model to keep our implementation as simple as possible and have a common baseline for a fair comparison. Furthermore, in the resampling process of AMCL, we also consider the buildings, map borders, and the GNSS uncertainty regions: If a particle is updated to a location inside a building or outside the map, we do not resample from this particle so that it does not survive in the next iteration step. Furthermore, we also consider GNSS information in the same manner as we do in our approach by only considering



(a) Driven distance: 0 m. (b) Driven distance: 15 m.

(c) Driven distance: 90 m. (d) Driven distance: 290 m.

(e) Driven distance: 1020 m. (f) Driven distance: 2840 m.

Figure 8.14: Sequence of localization results of the interval method at different time steps for trajectory 0027. All feasible poses of all rotation bins are combined into each figure. The sizes of the rotated box are shown in Figure 8.13a.

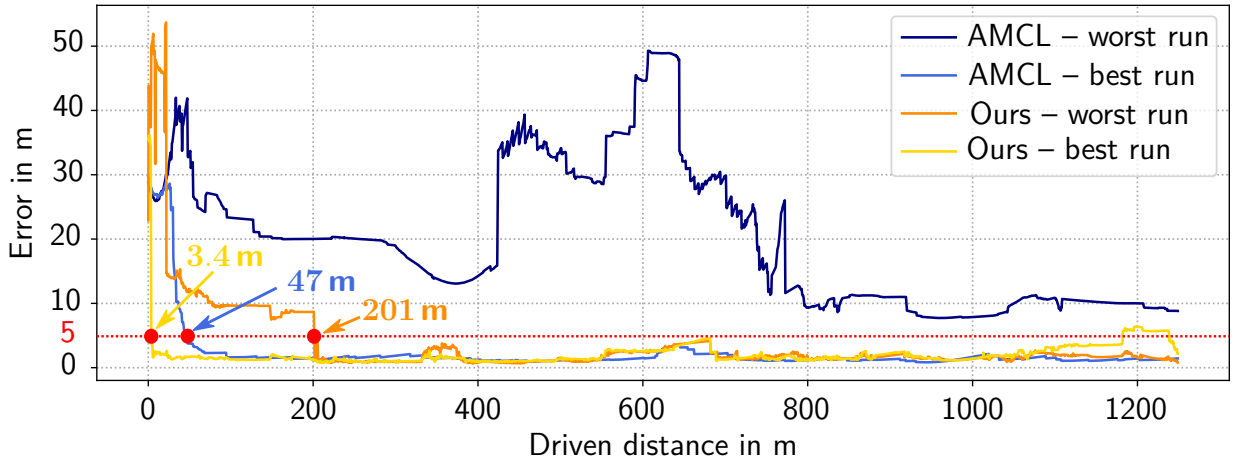


Figure 8.15: Average particle distance to ground truth for trajectory T_1 for the best and worst runs of our method and AMCL among 20 runs. The convergence points are marked.

those particles inside the uncertainty region of the global position acquired by the GNSS sensor. We are choosing the same uncertainty region for AMCL as we are using in our method.

When particles get depleted in the AMCL, we uniformly spread new particles in a radius of $100\,\text{m}$ around the actual position with random orientation. To ensure a representative comparison between the methods, we run AMCL and our hybrid method for each dataset 20 times. The convergence to the correct pose quantifies the performance of localization methods. However, to measure the convergence, we need to take, on the one hand, the convergence speed and, on the other hand, the tracking behavior of the correct pose after convergence into account. We define the convergence point as the driven distance after which the average particle position drops below $5\,\text{m}$. We consider the tracking error after convergence by analyzing the average estimation error. While Table 8.6 lists the results for the convergence analysis for all datasets, Table 8.7 shows the tracking results for all datasets. To obtain a more detailed overview of the localization behavior of our method compared to AMCL, we also plot the average particles distance to ground truth along the whole trajectory of T_1 in Figure 8.15 and of 0027 in Figure 8.16. Therefore, we consider two runs among the 20 for T_1 and 0027 for which we observed the fastest and slowest convergence.

Convergence Speed and Repeatability

As shown in Figure 8.15, while the worst run with our hybrid approach converges at $201\,\text{m}$, the best run already at $3.4\,\text{m}$. In contrast, the best run with the AMCL converges at $47\,\text{m}$. While AMCL's best run converges faster than the worst run of our hybrid approach, the worst run of AMCL does not converge at all. Taking the first row of Table 8.6 into account, which considers all 20 runs for T_1, we found that for this particular trajectory, our method converges on average 4.7 times faster than AMCL. Furthermore, according to Table 8.6, the average particle distance after convergence is lower with our approach than the AMCL.

We observe a similar result for 0027 in Figure 8.16. While the best run of our method already converges at $30\,\text{m}$, the worst run converges after AMCL's best run. According to

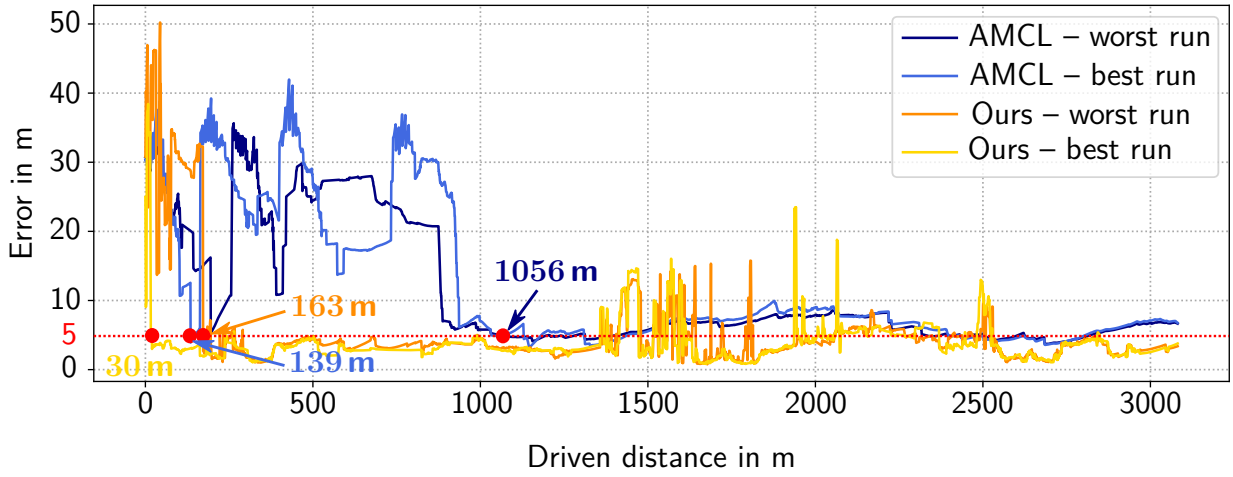


Figure 8.16: Average particle distance to ground truth for trajectory 0027 for the best and worst runs of our method and AMCL among 20 runs. The convergence points are marked.

Table 8.6, our method converges on average 3.7 times faster than AMCL for the trajectory 0027. Furthermore, AMCL's average particle error in Figure 8.16 for the best run jumps shortly after falling below the $5\,\text{m}$ limit to an error above $30\,\text{m}$. This is due to the tracking behavior we will analyze in the next paragraph.

Regarding the convergence behavior, we obtain similar results for other trajectories as summarized in Table 8.6. As the first column shows, our hybrid method converges on average faster than AMCL except for 0034. This does not mean that the AMCL always converges slower. As shown in Figure 8.15 and Figure 8.16, a fortunate resampling in the AMCL method may lead to a faster convergence for specific runs. However, on average, the AMCL shows slower convergence. In the worst case, the AMCL does not converge as shown in the third column of Table 8.6 while our method always converges to the correct solution for almost all of the datasets. For 0020, AMCL converges in none of the 20 runs. Only for 0034 and 0034^*, our method does not always converge since both datasets are recorded in residential areas where trees and hedges occlude significant parts of the buildings. This is also why our methods and AMCL converge very lately for both datasets. Another observation model may improve the result for 0034 and 0034^*.

The main reason for the faster convergence of our hybrid method is the interval method that reliably keeps track of the feasible region and enables an aggressive resampling process for the bounded MCL. While the interval method implicitly stores the past trajectory information by providing the feasible set of poses at each time step, the AMCL implicitly reflects this information in the set of particles. That is why AMCL performs a more defensive resampling so that a large variety of particles is kept as long as possible. However, due to unfortunate resampling, incorrect particles may be overpopulated, leading to particle depletion later. As the information on the feasible region is not available for AMCL, particles need to be spread randomly, ignoring the already driven trajectory – the convergence speed suffers. The aggressive sampling process in our hybrid approach continuously checks for better solutions in the feasible region so that switching to the correct solution is possible. On the downside, the aggressive

Dataset	Average driven distance until convergence		Standard deviation of driven distance until convergence		Runs with average error below 5 m (at least once)	
	Ours	AMCL	Ours	AMCL	Ours	AMCL
T_1	**80.12** m	377.58 m	**74.95** m	230.78 m	**100** %	95 %
T_2	**163.78** m	305.26 m	**112.94** m	168.28 m	**100** %	75 %
0018	**152.29** m	653.08 m	**9.6** m	361.53 m	**100** %	85 %
0020	**263.34** m	-	**51.66** m	-	**100** %	0 %
0027*	**107.2** m	406.95 m	**25.23** m	181.41 m	**100** %	60 %
0027	**98.57** m	365.76 m	**38.71** m	287.1 m	**100** %	**100** %
0028	**526.2** m	1632.79 m	**145.68** m	277.89 m	**100** %	**100** %
0033	**694.42** m	952.36 m	**129.69** m	184.77 m	**100** %	65 %
0034*	**443.17** m	592.77 m	**75.03** m	178.95 m	**90** %	20 %
0034	2364.47 m	**2283.24** m	**723.68** m	1014.5 m	**95** %	30 %
0071	**148.62** m	226.98 m	1.74 m	**0.0** m	**100** %	5 %

Table 8.6: Convergence evaluation. Both approaches are run 20 times on each data set. The average and standard deviation of the driven distance until convergence are only considered for those runs that converge. Those runs are classified as runs with average particle error below 5 m (at least once).

resampling leads to less accurate tracking results, as it is shown Figure 8.16, which will be analyzed in the next paragraph.

To study the repeatability of the methods, we also compare the standard deviation of the driven distance until convergence. The results are depicted in the second column of Table 8.6. The standard deviations of the driven distances are significantly lower with our method, revealing that the convergence distances are more similar among all runs except for 0071. The exception is because AMCL only converged once (5 %) among 20 runs (third column). The main reason for the higher repeatability of our method compared to the AMCL is the deterministic interval method. As the interval approach always provides the same result on the feasible region, the bounded MCL – the random component in our hybrid approach – is restricted to this deterministic region. The AMCL has no deterministic components – so each run can lead to totally different results.

Tracking Error After Convergence

Up to now, we considered the convergence speed. As we define the convergence speed as the driven distance at which the average particle error drops below 5 m, the question remains how well do the localization methods track the correct pose after the convergence point so that the localization error stays small. Therefore, let us consider again the average particle errors along the trajectory shown in Figure 8.15 and Figure 8.16. While in Figure 8.15, our method and the AMCL can keep the error comparatively low after convergence, at 1200 m the run with the fastest convergence of our method (yellow) exceeds the 5 m error limit. Hence our method could not keep the average particle error below the defined limit, while the AMCL does not converge at all for the worst case. Figure 8.16 illustrates the problem more clearly.

Neither our method nor AMCL is able to keep the average particle error below 5 m after the convergence points. However, we observe different mechanisms for the methods that lead to larger errors, We will analyze them in the following.

Let us first focus on the AMCL. The AMCL's average particle error jumps shortly after falling below the 5 m limit to an error above 30 m at 139 m driven distance, as mentioned above. This is caused by particle depletion: We experienced in our experiments that in the AMCL, particles close to the true pose survive longer – however, if the orientation of the remaining particles is slightly off, the particles will at some point be deleted as they become infeasible. To counteract this problem, we could spread more particles. However, increasing the number of particles leads to longer runtimes. Furthermore, the selection of how many particles need to be spread depends on the map and the trajectory. Hence, although the localization with AMCL may work for this dataset when we increase the number of particles, this does not guarantee that the number of particles is sufficient for another dataset.

Furthermore, from 900 m onward, the AMCL average particle error is significantly lower than before. Nonetheless, the AMCL is not able to further reduce the error. The main reason for this behavior is that AMCL needs to keep track of many particles to prevent particle depletion. This also means that the average particle estimate will consider low-weighted particles leading to more significant errors with this metric.

In contrast, our method can significantly reduce the average particle error. However, our method shows another unfavorable behavior: Error spikes at around 1500 m in Figure 8.16 occur. The main reason for the spikes is the constant checking for better solutions inside the feasible set: Particles may populate regions enclosing temporarily better solutions. While the aggressive resampling procedure leads to fast convergence, it comes with the cost that good solutions may be omitted earlier as, temporarily, other parts of the feasible set may fit better to the LiDAR data. Due to the aggressive resampling, the better particle is preferred, leading to an error spike. Hence, our method is more sensitive to cases where the observation model does not properly weight the particles. This is also why spikes are clustered between 1300 m and 2100 m. In this trajectory part, the building walls are occluded by trees and hedges. While our method generates error spikes, the AMCL keeps the average particle error at an almost constant but high error level.

For the other trajectories, we observe similar results. The results are summarized in Table 8.7. According to the first column, the average particle distance to the correct pose is lower for our hybrid method except for 0028, 0034, and 0071, although the error spikes occur. The problems with 0034 and 0071 are already mentioned above. However, 0028 again shows the weakness of our approach. The aggressive resampling leads to larger jitter of the localization estimate compared to the AMCL, although our method converges 3 times faster to an error below 5 m according to Table 8.6. We measure the jitter based on the standard deviation of the average particle distance in the second column of Table 8.7. The smaller the standard deviation, the smaller the jitter of the average particle error. For 0028, we can see that the standard deviation is much lower for the AMCL than our method. However, this is not the case for all trajectories. Depending on the maps and the trajectories, the methods provide different results. For instance, for T_1, AMCL shows preferable behavior as it provides more stable estimates, although the average particle error with the AMCL is larger compared to our

Dataset	Average particle error after convergence		Standard deviation of particle error after convergence	
	Ours	AMCL	Ours	AMCL
T_1	**1.99** m	2.32 m	1.24 m	**0.96** m
T_2	**1.91** m	2.15 m	2.32 m	**1.12** m
0018	**3.51** m	9.82 m	**2.79** m	7.68 m
0020	**4.24** m	-	**3.66** m	-
0027*	**4.79** m	5.06 m	7.13 m	**3.67** m
0027	**3.79** m	10.24 m	**2.44** m	8.14 m
0028	11.93 m	**5.89** m	12.27 m	**1.48** m
0033	**7.95** m	12.22 m	**5.08** m	8.16 m
0034*	**7.82** m	8.68 m	**4.35** m	7.91 m
0034	18.17 m	**15.01** m	10.55 m	**7.08** m
0071	4.5 m	**3.3** m	4.33 m	**0.38** m

Table 8.7: Tracking evaluation. Both approaches are run 20 times on each data set. The average particle error after convergence and the dtandard deviation of the particle error after convergence is shown.

method. However, for 0027, our method has less jitter in the localization estimate than AMCL. If we consider Figure 8.16, we can see that our method mainly has temporary error spikes after convergence. In contrast, the AMCL has larger parts where the average particle error changes rapidly due to particle depletion.

In summary, our method converges faster than the AMCL. However, our method is more sensitive to problems caused by the observation model, due to which the tracking performance suffers. Hence, more sophisticated observation models are vital for the good performance of our method. Nonetheless, our method constantly checks better solutions inside the feasible set so that tracking losses lead to error spikes that are corrected rapidly. The AMCL is more robust concerning problems of the observation models. However, the error stays comparatively large since the AMCL has to consider a large variety of particles. This hinders the average particle error from being further reduced.

8.2.3 Runtime

While the average operation times along the whole trajectories are presented in the first column of Table 8.8, Figure 8.17 and Figure 8.18 show the average operation times per frame depending on the driven distance for T_1 and 0027, respectively. The LiDAR data was acquired at 10 Hz for all datasets. As a result, a real-time operation requires an operation time below 0.1 s per frame. According to the first column in Table 8.8, our method satisfies the real-time requirement for all datasets with the average operation time per frame. The AMCL does not fulfill the runtime requirement for all datasets. In the worst case, the AMCL needs in 0028 an average operation time per frame up to three times longer than real-time. In contrast, our method only needs a third of the available time on average for the same dataset.

Dataset	Average operation time per frame		Standard deviation of operation time per frame	
	Ours	AMCL	Ours	AMCL
T_1	0.038 s	$\mathbf{0.025}$ s	$\mathbf{0.041}$ s	0.077 s
T_2	$\mathbf{0.033}$ s	0.034 s	$\mathbf{0.022}$ s	0.079 s
0018	$\mathbf{0.034}$ s	0.071 s	$\mathbf{0.02}$ s	0.33 s
0020	$\mathbf{0.03}$ s	0.071 s	$\mathbf{0.023}$ s	0.076 s
0027^*	$\mathbf{0.067}$ s	0.188 s	$\mathbf{0.043}$ s	0.68 s
0027	$\mathbf{0.043}$ s	0.18 s	$\mathbf{0.027}$ s	0.63 s
0028	$\mathbf{0.034}$ s	0.292 s	$\mathbf{0.017}$ s	0.38 s
0033	$\mathbf{0.036}$ s	0.173 s	$\mathbf{0.023}$ s	0.57 s
0034^*	$\mathbf{0.051}$ s	0.106 s	$\mathbf{0.045}$ s	0.21 s
0034	$\mathbf{0.062}$ s	0.113 s	$\mathbf{0.038}$ s	0.42 s
0071	$\mathbf{0.034}$ s	0.068 s	$\mathbf{0.025}$ s	0.09 s

Table 8.8: Average operation time per frame and the standard deviation of the operation time per frame.

The standard deviations of the average operation times are listed in the second column of Table 8.8. Note that our method's standard deviation of the average operation time per frame is significantly lower. Hence, the runtime stays comparatively constant compared to the AMCL. Figure 8.17 and Figure 8.18 underline this observation based on the exemplary trajectories T_1 and 0027. While our method shows similar behavior in both plots, the AMCL operation time profile differs significantly. When the localization is initialized, the uncertainty is substantial. Hence, our approach has to consider a vast feasible region, while AMCL has to spread particles to cover the whole map widely. Consequently, for both methods, the initialization requires significant operation times. That is why both methods do not satisfy the real-time requirement initially. However, as our method successively dismisses infeasible subsets rapidly, the computational burden is gradually decreased so that our method reaches low operation times per frame. Nonetheless, there are rare frames at which the operation time exceeds the time limit of 0.1 s in Figure 8.17 and Figure 8.18. Hence, we cannot ensure hard real-time. Note that our machine uses no real-time kernels, so we cannot guarantee hard real-time execution.

Although the plots for the AMCL operation times in Figure 8.17 and Figure 8.18 significantly differ, in both plots, we can see noticeable jumps. Those jumps are caused by particle depletion: When all particles vanish as the remaining particles are updated to infeasible regions, new particles must be spread. We uniformly spread 20000 particles in a defined region as described above. As the number of particles suddenly rises, the operation time increases accordingly. As in Figure 8.17, the AMCL can rapidly reduce the number of particles to a small amount. The operation time for AMCL is comparatively low, except for those frames where new particles are spread. However, for the dataset 0027, the AMCL cannot aggressively reduce the number of particles. That is the reason why the operation time stays comparatively large. Due to the large number of particles that AMCL tracks, the operation time is higher compared to our approach. The performance of the AMCL is mainly dependent on the map and trajectory, while

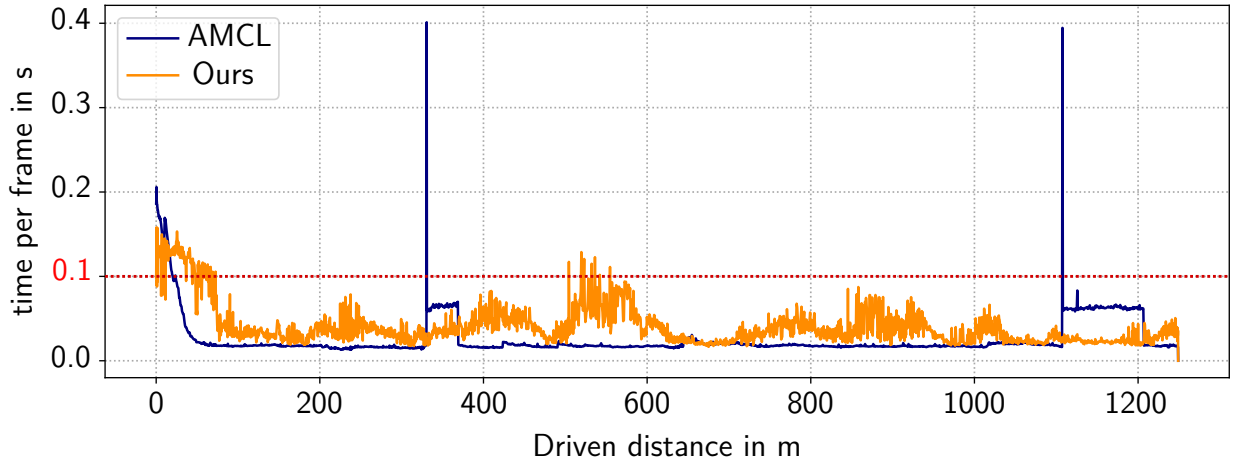


Figure 8.17: For each frame in trajectory T_1 the operation time is averaged among all 20 runs. The average operation time per frame is plotted for the driven distances.

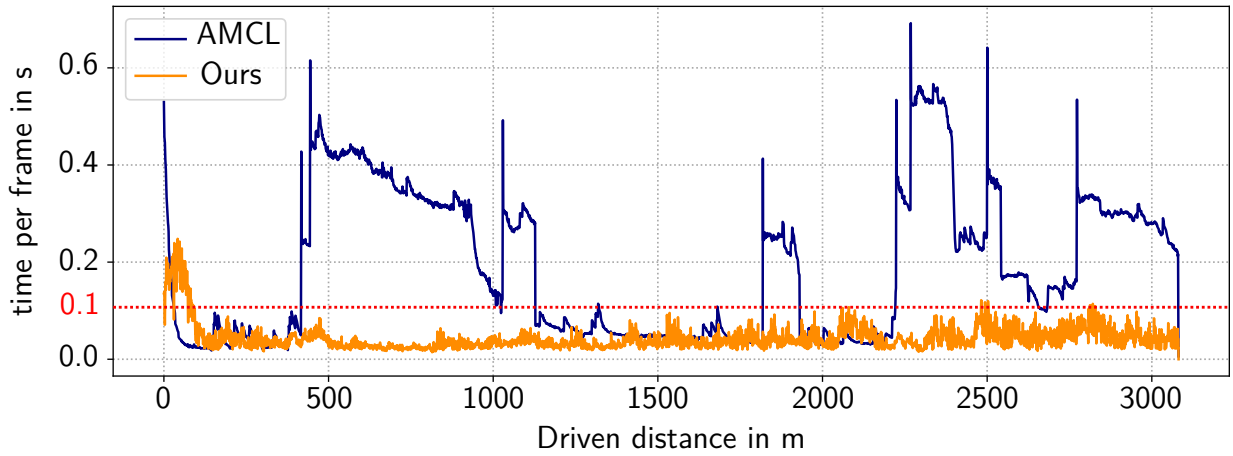


Figure 8.18: For each frame in trajectory 0027 the operation time is averaged among all 20 runs. The average operation time per frame is plotted for the driven distances.

our method performs similarly for all datasets regarding the operation time. As the number of particles changes suddenly for AMCL, the operation time per frame also changes suddenly, increasing the standard deviation of the average operation time per frame shown in the second column of Table 8.8.

Note that the online capability of our method depends on the size and structure of the map, the size of the initial pose estimate, and the number of cores available on the CPU. For both methods, the operation time before convergence takes longer than after. Nonetheless, our hybrid method is real-time capable for the tested scenarios. In contrast, the AMCL does not reach real-time performance for most KITTI datasets.

8.3 Refined Localization

Now we will turn our attention to the refined localization. In the HyPaSCoRe Localization, the hybrid refined localization module uses the coarse localization results to improve and refine

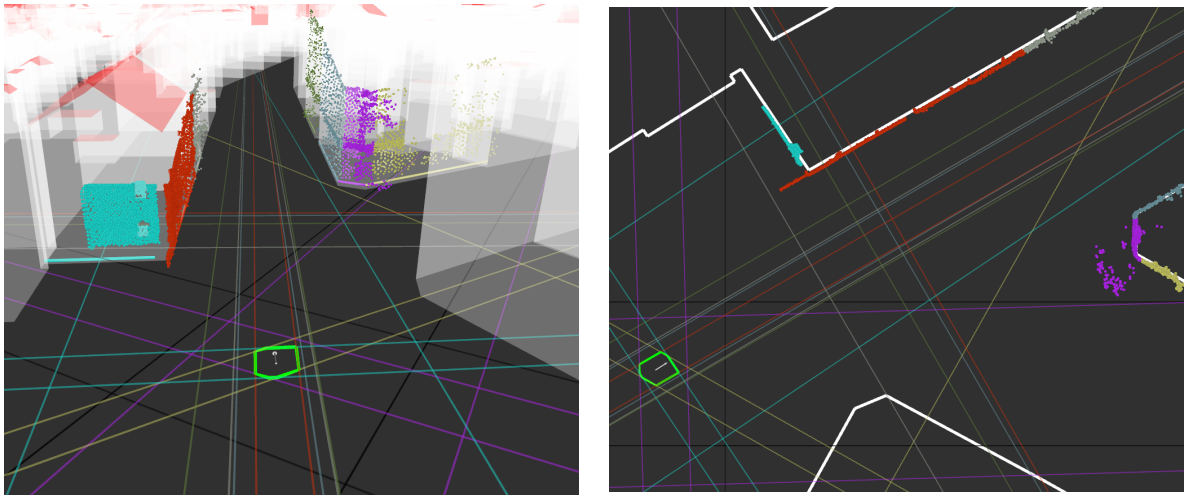
the pose estimate by performing local measurement association to facades in the map. In the scope of this section, we want to evaluate this module independently.

To achieve an independent operation of the refined localization, we need to introduce some minor modifications to the pipeline for the sake of the ablation study. Since the main objective of the coarse localization is to narrow down the feasible set of poses from a large set of possible poses, the coarse localization performs the major part of its work during initialization when the uncertainty is large. As we want to design our experiments so that the coarse localization has no impact on the pose estimation, we manually fix the initial pose uncertainty to a smaller region using the ground truth pose so that the refined localization is directly applicable. We also limit the maximum possible error during the trajectory to a smaller region. Theoretically, this experimental setup is appropriate for the ablation study. However, further investigation on the ground truth pose shows that for some KITTI trajectories, the OSM and the provided ground truth trajectories have a varying offset. Unfortunately, we cannot reconstruct the source of the error, whether the ground truth trajectory or the OSM are error-prone, and compensate for that error. On the bright side, for the author-collected datasets T_1 and T_2, the ground truth trajectories are consistent with the maps.

Nonetheless, we suggest the following to use the KITTI dataset for evaluation. Along the trajectory, we provide for each frame a feasible set of poses that we define based on an appropriate region around the ground truth pose. The radius of the inflation depends on the maximal error that we determine empirically for each dataset beforehand. Similar to the bounded MCL in the coarse localization, we spread particles in that small inflated region to determine the best pose, which we use to initialize the refined localization. By doing this, we bypass the ground truth problem of the KITTI dataset coming with the cost of a possibly sub-optimal performance of the refined localization as we cannot assure the best particle to be indeed an acceptable initial solution. While we choose for 0020 and 0034 an inflation radius of $10\,\mathrm{m}$ and for 0028 $20\,\mathrm{m}$, for all remaining six trajectories, we select a position inflation radius around the ground truth position of $5\,\mathrm{m}$. For the ground truth angle inflation, we select for all datasets an inflation radius of $5°$.

In Subsection 6.2.2, we introduced a set-membership-based and a probabilistic association. Here we only evaluate the probabilistic association that chooses the best particle among all particles in the feasible set, as this association method is used in the HyPaSCoRe Localization pipeline. The results on the set-membership-based association can be found in [143].

The refined localization consists of two parts. In the first part, based on the points-to-facades association, the consistent set of poses is determined by computing a polygon that describes the set of consistent positions and an interval that encloses the consistent set of orientation angles. In the second part, we apply a bounded optimization approach to determine the most likely solution within the consistent set. Accordingly, we also divide our evaluation into two parts. In the first part in Subsection 8.3.1, we evaluate the pessimism of the set-membership-based uncertainty estimation. Therefore, we study the sizes of the obtained sets and analyze the association. We evaluate the sets' width corresponding to the polygons' widths and the rotation angle intervals. Suppose we admit an inflation of r for the position. In that case, the maximal side length of the rotated rectangle can be $\sqrt{2}r$, which is the size of the diagonal of the maximally permitted uncertainty region. Note that during the regular operation of the



(a) Many differenly oriented facades – 3D view. (b) Many differenly oriented facades – 2D view.

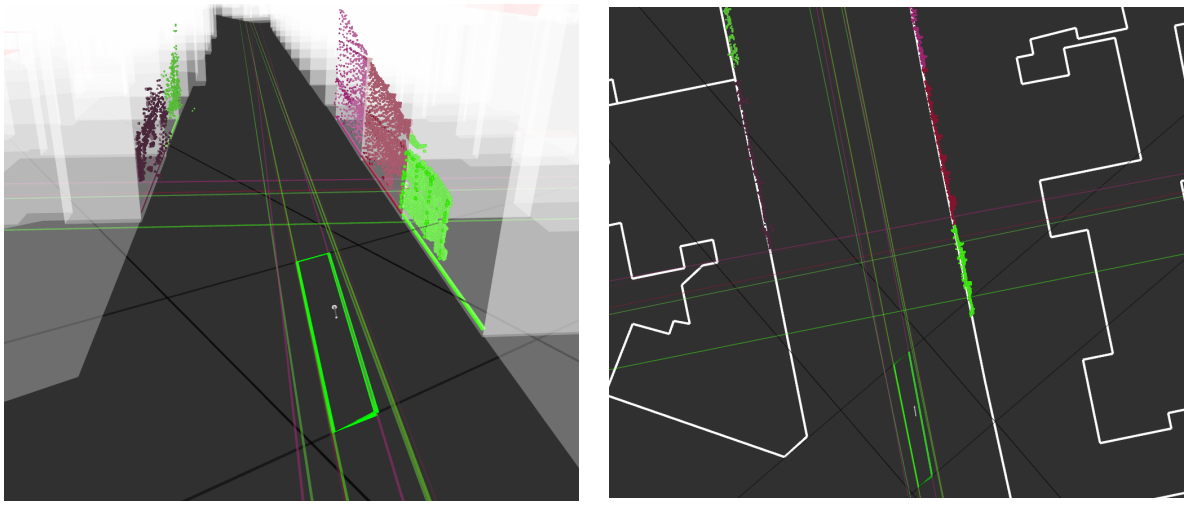
Figure 8.19: An exemplary configuration with many facades. The interval-based localization result is shown in the 3D view (Figure 8.19a) and the 2D view (Figure 8.19b). The extracted facade points using the iHT and the stripes corresponding to the facades are colored accordingly.

HyPaSCoRe Localization, the region in which the refined localization operates is restricted by the coarse localization. The maximum error for the orientation is $10°$, twice the rotation angle inflation radius. In the second part in Subsection 8.3.2, we evaluate the bounded optimization results and compare those with a classical unbounded MLE approach. Therefore, we can use classical evaluation metrics to compare the most likely poses with the ground truth.

The performance of our refined localization approach mainly depends on the map's topology. Although we evaluate our method on nine different datasets and provide condensed results based on common evaluation metrics, we want to provide a more in-depth evaluation on two of nine dedicated, exemplary datasets. First, we choose T_1 with the LOD2 map, which has higher accuracy, and second, 0027 with the OSM, which has lower accuracy. Both maps are topologically different, as we will see later. The uncertainty of building maps is determined by the position and orientation uncertainty of the facades. For the LOD2 maps, we account for a position uncertainty of up to $1\,\mathrm{m}$ and an orientation uncertainty of up to $1.7°$. For the OSMs, we account for $1.3\,\mathrm{m}$, and $2.9°$. The uncertainty bounds are estimated empirically.

8.3.1 Set-Membership-based Localization

In Figure 8.19, the set-membership-based localization result for a scene with many differently oriented facades is presented. While Figure 8.19a shows the localization result in 3D, Figure 8.19b shows the result in 2D. The associated points to the different facades are colored. Note that those seen facades are marked with a line in the specific color in which the associated points are displayed. Each of the seen facades generates position stripes that we illustrated with an upper and lower border line on the ground plane. The stripes are accordingly colored. The intersection region of all stripes that represents the consistent set of positions of the robot is highlighted with a green polygon. A building geometry as presented in Figure 8.19 is favorable for the proposed localization scheme since many differently oriented facades restrict



(a) Similarly oriented facades – 3D view.

(b) Similarly oriented facades – 2D view.

Figure 8.20: An exemplary configuration with facades with similar orientation. The interval-based localization result is shown in the 3D view (Figure 8.20a) and the 2D view (Figure 8.20b). The extracted facade points using the iHT and the stripes corresponding to the facades are colored accordingly.

the pose along many directions. Consequently, we obtain many differently oriented stripes that lead to a small intersection region.

In contrast to a good constellation of the building facades, Figure 8.20 shows a less favorable building constellation that leads to poor localization results with our localization scheme. All facades are similarly oriented, due to which all stripes are similarly oriented. In our experiments, we restricted the initial set of positions to $5\,\mathrm{m} \times 5\,\mathrm{m}$ rectangle. Only this initial restriction further contracts the positions in the parallel direction. Note that we also obtain constraints for the position in the parallel direction due to the line segment intersection constraint. However, the constraints are less restrictive in this situation than the initial position constraint.

To illustrate the impact of the building geometry in the vehicle's vicinity on the localization estimation, we display the uncertainty of the localization estimate along the small trajectory part of T_1 in Figure 8.21. To evaluate the localization performance of our method, we need to quantify the accuracy. The radius of the interval can measure the orientation accuracy. To measure the size of the position polygon, we fit an enclosing rotated rectangle with the smallest area. The width and the height approximate the position accuracy. Figure 8.21 displays the position polygons for a small part of the trajectory T_1 where each polygon is colored based on the length of the largest side length of the rotated enclosing bounding box. While blue encodes a comparatively high uncertainty, red highlights small uncertainty. Note that in T_1, the front-facing Cepton LiDAR-sensor was used. Due to the LiDAR's limited observation cone, only those poses from which facades with a perpendicular orientation to the driving direction are seen have lower uncertainty. The localization estimate becomes uncertain if the LiDAR only captures very similarly oriented facades.

In the following, we will consider the two datasets T_1 and 0027 in more detail to evaluate the sizes of the computed consistent set.

Figure 8.21: Largest side lengths of the localization position polygons.

8.3.1.1 Author-Collected Dataset T_1 with LOD2 map

The overview to the trajectory and the map is visualized on in middle right in Figure 8.22. We also show close-up figures with the determined polygon (blue) and the enclosing rotated rectangle (orange) for dedicated driven distances. The driven distances to which the localization polygons refer are denoted in green in the close-up figures. To evaluate how the polygon size and rotation interval width change along the trajectory, we also provide the side lengths of the rotated rectangle for the driven distances and the rotation interval widths in Figure 8.23a. The sample close-ups in Figure 8.22 are accordingly marked in Figure 8.23a by a red dashed vertical line. To measure the size of the position polygon, we wrap the smallest rotated box that minimizes the area. We measure the polygon's size by considering the long and short sides of the rotated rectangle. We plot the interval width for the orientation in Figure 8.23b.

When we compare the progression of the long side length (blue) and the short side length (orange) in Figure 8.23a, it is noticeable that the long side length changes to a greater extent, while the short side length always stays around $2\,\mathrm{m}$ with an average of $1.57\,\mathrm{m}$ according to Table 8.9. A closer inspection of the long side length in Figure 8.23a reveals a repeated saw-tooth pattern where the long side length continuously rises and suddenly shrinks to a small value. For instance, this pattern is particularly strong between $400\,\mathrm{m}$ and $560\,\mathrm{m}$. If we compare this portion in the trajectory overview in Figure 8.22, this part corresponds to the top left part where the vehicle is between buildings in a sluice-like street so that it can only see buildings on the left and right side. Consequently, since in this part, no facades are seen perpendicular to the driving direction, the uncertainty along the driving direction rises continuously. That is why the polygon size rises along the driving direction as illustrated in the close-up in Figure 8.22: Initially at $400\,\mathrm{m}$, the polygon is comparatively small. However, at $480\,\mathrm{m}$, the polygon shape gets more elongated in the driving direction. Considering Figure 8.23a, at $480\,\mathrm{m}$, the vehicle has evolved almost half of the sluice-like portion of the trajectory. Shortly after $540\,\mathrm{m}$ of driven distance, the large side length jumps to a smaller value due to observing a perpendicular facade that makes the contraction of the consistent set possible. If we consider Figure 8.22 at $560\,\mathrm{m}$,

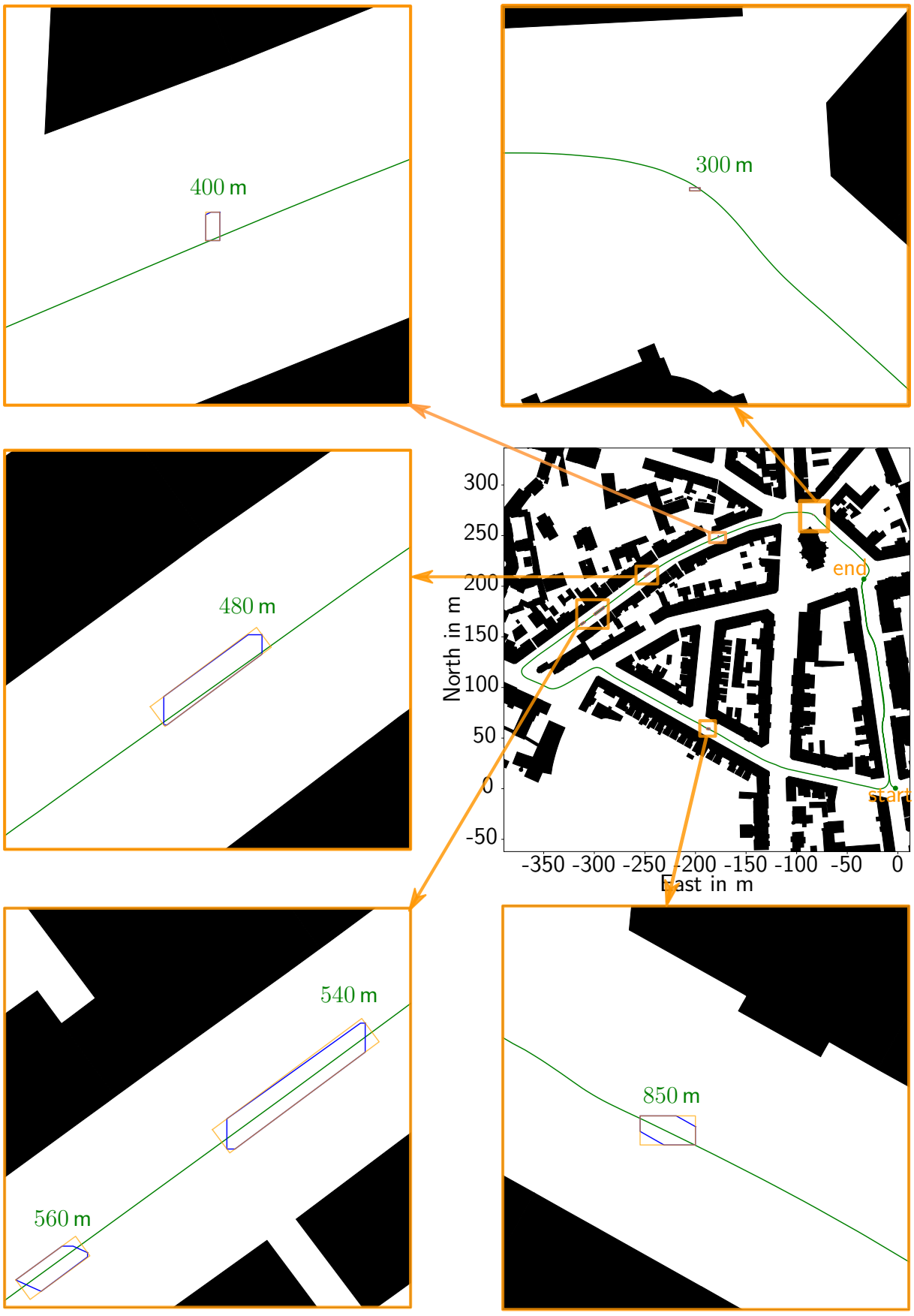
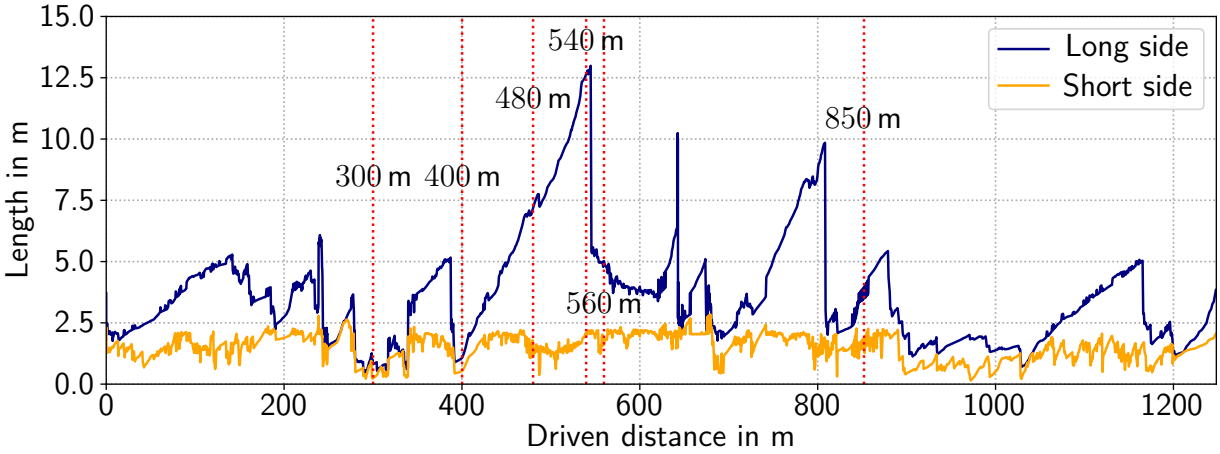
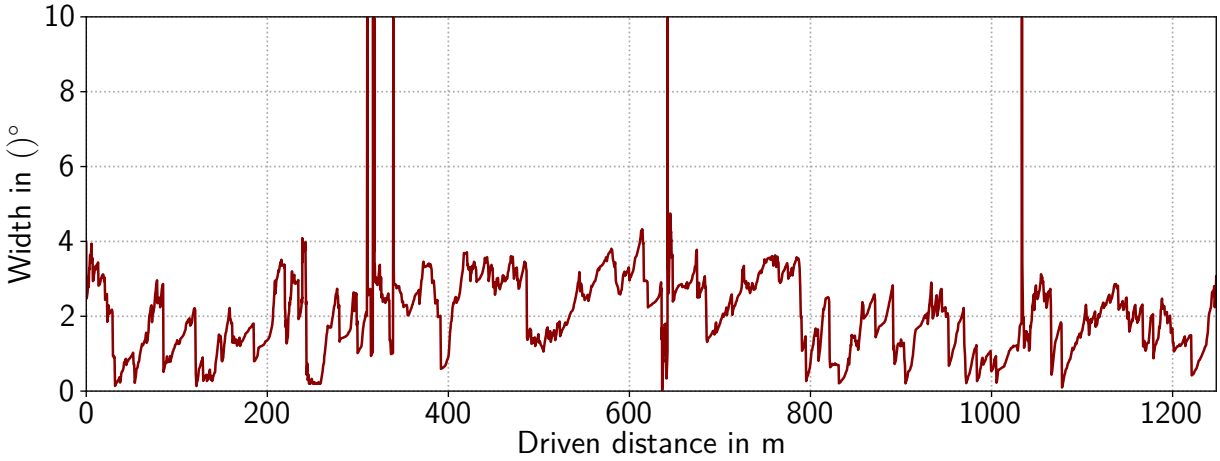


Figure 8.22: T_1 trajectory overview and selected pose estimation. Polygons are colored blue, the rotated rectangle is colored orange.



(a) Side lengths of the rotated box that encloses the consistent polygon set of positions at each frame.

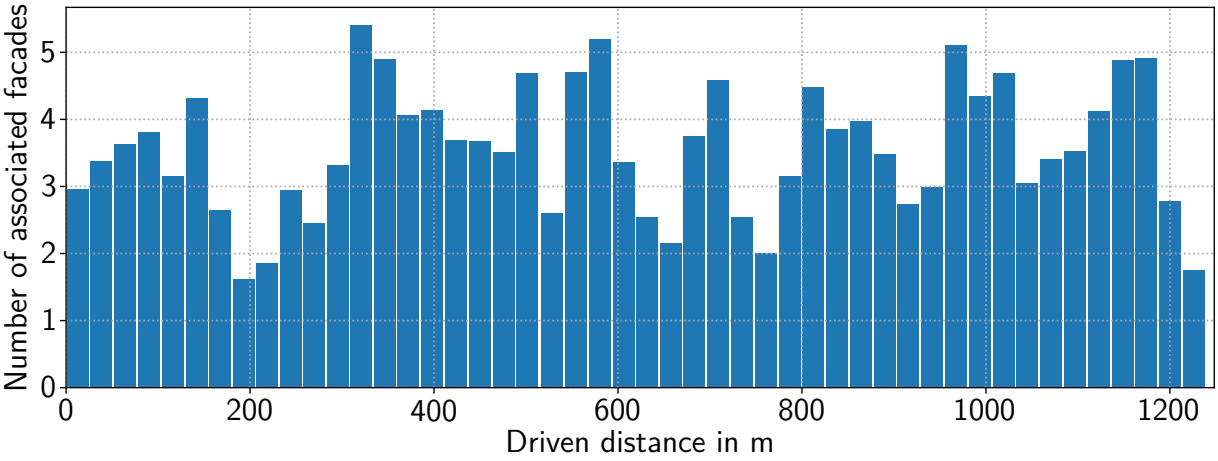


(b) Width of the orientation interval that encloses the consistent set of orientations.

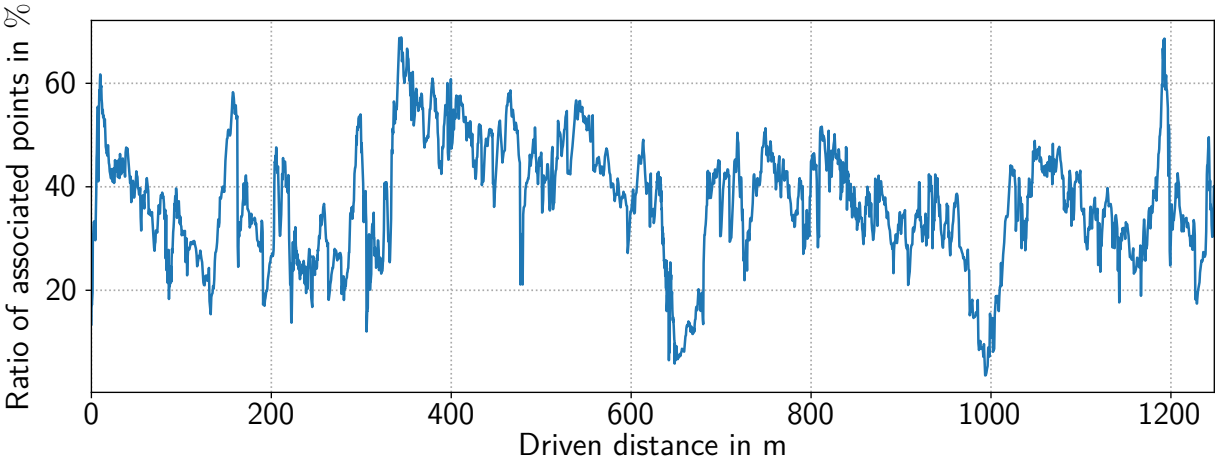
Figure 8.23: Sizes of the enclosing rotated rectangle for the position (Figure 8.23a) and the orientation (Figure 8.23b) for trajectory T_1 using the LOD2 map. The sizes represent the uncertainty of the polygon pose estimation. The according polygons and enclosing rotated boxes are visualized for selected frames in Figure 8.22. The frames are highlighted by red dashed lines in Figure 8.23a.

the vehicle associates its local LiDAR measurements to a facade at the end of the road. Hence, the uncertainty in the driving direction is reduced. The contracted position polygon is depicted in the bottom left close-up in Figure 8.22.

The rotation interval width shown in Figure 8.23b is almost independent of how many facades are seen. As already one facade observation provides good contraction on the rotation interval, the interval width is mainly affected by the iHT estimation of the local line. Since the iHT performs a line parameter estimation, the quality of the results depends on how well the perceived points construct a line in the projected LiDAR scan. Consequently, the more points are associated with a facade, the better the local line estimates will be, and accordingly, the orientation interval width will shrink. Sometimes the interval width jumps to the maximal value of $10°$. This happens if the association is error-prone, so the orientation interval becomes empty. In that case, we assume maximal uncertainty for the orientation.



(a) Number of seen facades along the trajectory.



(b) Ratio of associated points among all LiDAR measurements along the trajectory.

Figure 8.24: Evaluation of the association. The number of seen facades (Figure 8.24a) and the ratio of associated LiDAR points (Figure 8.24b) affect the localization accuracy.

Figure 8.24 illustrates how well our method associates local LiDAR measurements to building facades. While Figure 8.24a plots the number of facades along the trajectory, Figure 8.24b depicts the ratio of how many LiDAR measurements from the total amount of measurements are associated with facades along the trajectory. The highest number of facades is seen at $300\,\mathrm{m}$. As shown in Figure 8.22, the vehicle is placed in a junction where many different oriented facades are observed. This is also the reason why at this driven distance, the polygon size ($< 1\,\mathrm{m}$) and orientation interval width ($< 1°$) is very small.

Nonetheless, shortly after the orientation interval is contracted to a very small interval, the interval width jumps to the maximal value. The main problem of an overly optimistic and small orientation interval is that already small perturbations may lead to empty sets if the iHT underestimates the uncertainty of the local line orientation. In the case of more significant occlusions, the association ratio drops. In Figure 8.24b, we see a ratio drop at $650\,\mathrm{m}$. According to Figure 8.22, the vehicle was located on a left turn in the left part of the map. During this turn, the LiDAR was occluded by other traffic participants and parked cars.

8.3.1.2 KITTI Dataset 0027 with OSM

The map we use for the 0027 KITTI dataset is available on the OSM database. Although OSM has less accurate building footprints, we can localize the vehicle using our refined localization method. Since the dataset was collected in a residential area in Karlsruhe, the map significantly differs from the map that we considered previously for T_1. While the map in T_1 shows a very dense structure with connected buildings (cf. Figure 8.22), the 0027 map as shown in Figure 8.25 contains many disconnected buildings that are partially occluded by for instance vegetation.

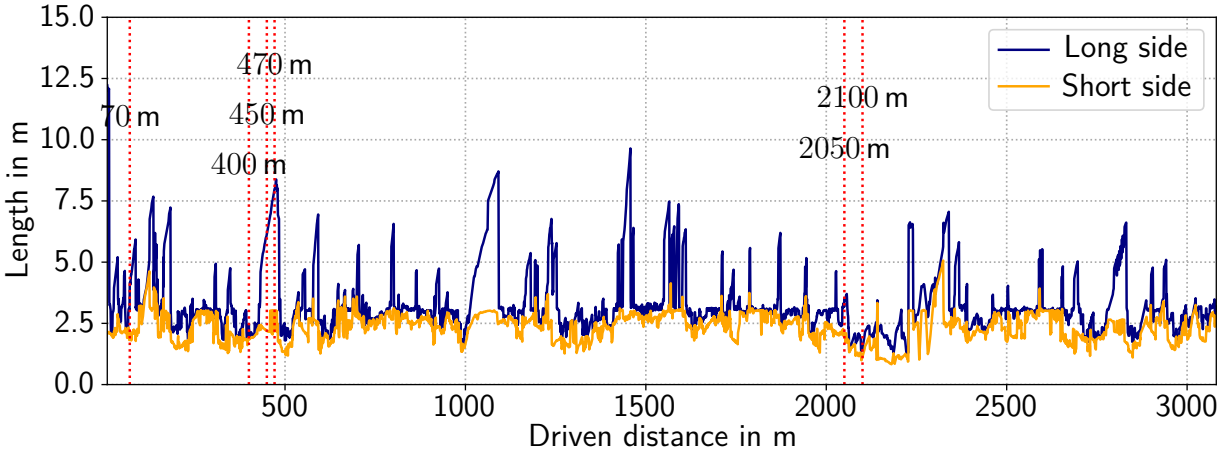
Similar to the overview Figure 8.22 for T_1, the overview of the whole trajectory and dedicated close-ups on the set-membership-based localization results are shown for the KITTI dataset 0027 in Figure 8.25. When directly comparing the close-ups in Figure 8.22 and Figure 8.25, it can be noticed that the polygons in Figure 8.25 do not contain the green ground truth trajectory path while the polygons in Figure 8.22 do. This is due to the KITTI dataset's error-prone ground truth trajectory. Although the relative motion of the vehicle is correct for the major part of the KITTI ground truth trajectories, the absolute localization accuracy reveals an offset to the OSM data. Particularly the bottom right close-up figure in Figure 8.25 shows the inconsistent offset of the OSM and the ground truth trajectory since the green ground truth trajectory touches the building so that the vehicle should have collided with the building. Nonetheless, the offset is within $5\,$m range, and therefore the ground truth can still be used as a rule of thumb in our evaluation. Due to this offset, the polygons do not directly contain the green path as is the case in Figure 8.22, where we have very accurate absolute ground truth.

Figure 8.26 shows the set-membership evaluation similar to the already introduced plots in Figure 8.23. Although the saw-tooth pattern is much stronger in T_1 (Figure 8.23a), in the 0027 trajectory between $400\,$m and $500\,$m and between $1000\,$m and $1100\,$m we can notice a similar behavior. As denoted by the red dashed vertical lines, the polygons for the saw-tooth pattern between $400\,$m and $500\,$m are depicted on the top left close-ups in Figure 8.25. In this portion of the trajectory, the vehicle is in a similar sluice-like street enclosed by a left and right row of connected buildings. This leads to a growing uncertainty along the driving direction until a perpendicular facade is detected. However, according to Table 8.9, the long side has an average length of $3.25\,$m for 0027 compared to $3.56\,$m in T_1 with the more accurate LOD2 map. This is mainly because 0027 provides more visible, differently oriented facades as more separate buildings are in the residential area. Nonetheless, the uncertainty of the OSM is higher compared to the LOD2 map due to which, according to the second column in Table 8.9, the short side length is much smaller with $1.57\,$m in T_1 compared to $2.3\,$m in 0027.

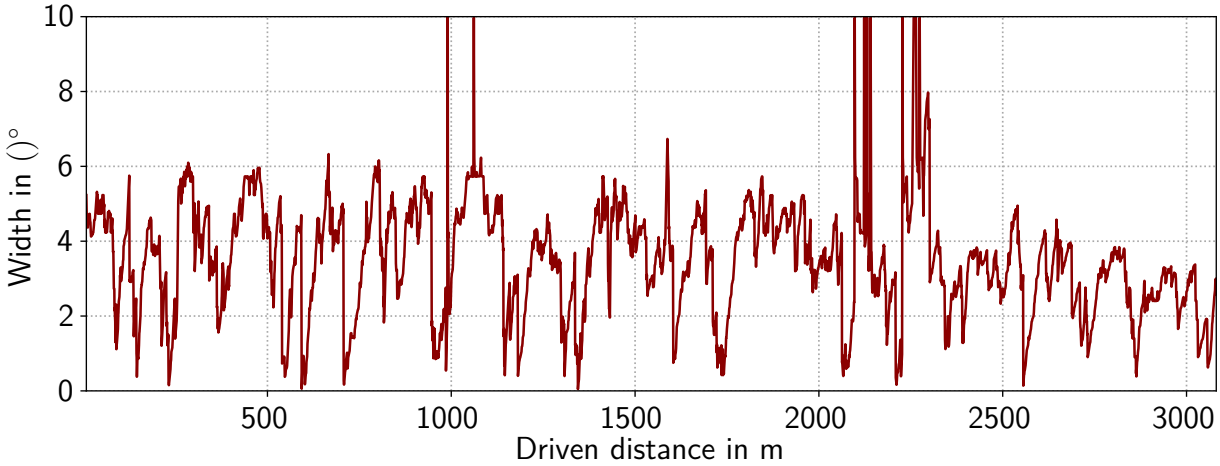
The rotation interval width along the trajectory is depicted in Figure 8.26b. Similar to Figure 8.23b, we also experience sudden jumps to the maximally permitted orientation error due to inconsistencies in the association that lead to empty sets. Furthermore, the angle uncertainty in Figure 8.26b is higher compared to Figure 8.23b as we account for the OSM higher uncertainties as explained above.

The association evaluation is depicted in Figure 8.27. Note, that in T_1 the association ratio reaches in Figure 8.24b more than $60\,\%$, while the association ratio in 0027 does not even reach $50\,\%$ according to Figure 8.27. The main reason for this observation is that different LiDAR systems are used. The KITTI dataset provides data of an HDL64 LiDAR with 360°

Figure 8.25: 0027 trajectory overview and selected pose estimation. Polygons are colored blue, the rotated rectangle is colored orange.



(a) Side lengths of the rotated box that encloses the consistent polygon set of positions at each frame.

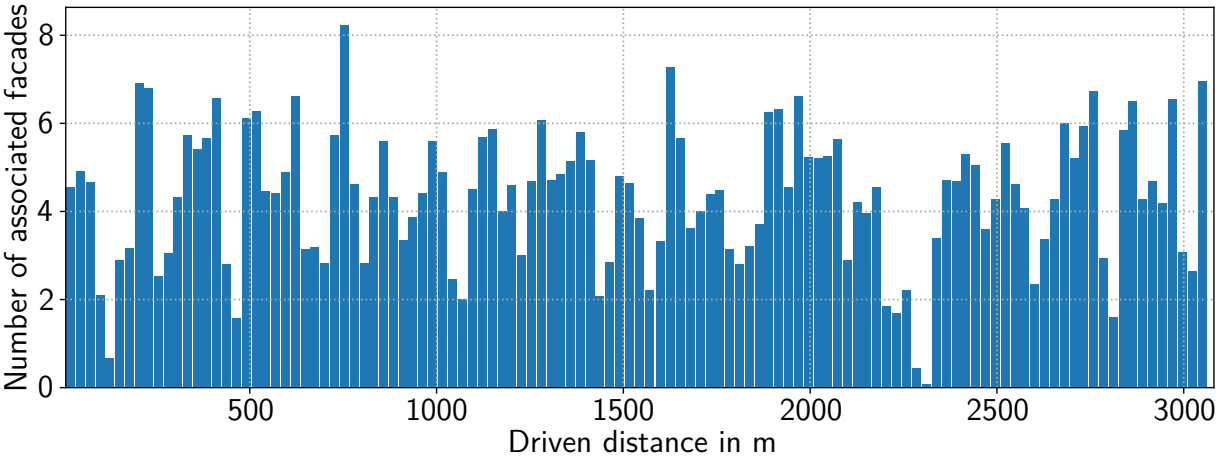


(b) Width of the orientation interval that encloses the consistent set of orientations.

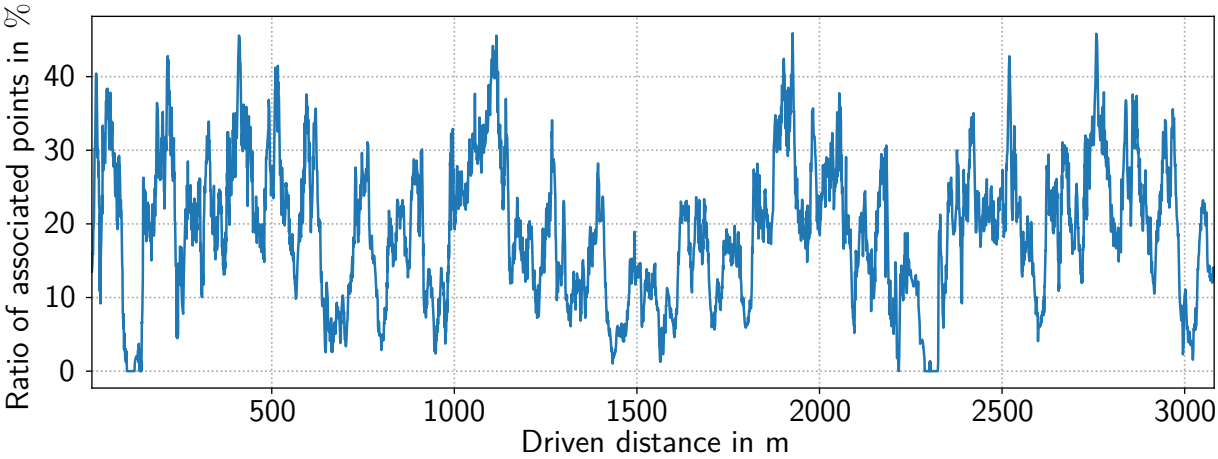
Figure 8.26: Sizes of the enclosing rotated rectangle for the position (Figure 8.26a) and the orientation (Figure 8.26b) for trajectory 0027 using the OSM. The sizes represent the uncertainty of the polygon pose estimation. The according polygons and enclosing rotated boxes are visualized for selected frames in Figure 8.25. The frames are highlighted by red dashed lines in Figure 8.23a.

FoV. In T_1, we utilize a front-facing solid-state LiDAR with $60° \times 40°$ FoV. In the case of the HDL64 LiDAR, a major portion of the scans measures the street, while the front-facing LiDAR scans perpendicular facades almost entirely. As a result, a larger part of the measured point of the front-facing LiDAR is more likely to measure points on a facade than with the HDL64.

Although it is not surprising, it is worth noticing that the number of seen facades is correlated with the ratio. For instance, at $2300\,\text{m}$ in Figure 8.27a almost no facades are seen for a short period, and accordingly, the ratio drops to zero in Figure 8.27b. The according portion of the trajectory in Figure 8.25 is the rightmost part where only on one side of the trajectory buildings are present. A closer inspection of this portion reveals that unmapped walls occlude the buildings, due to which no buildings are visible. This is also why the rotated rectangle side lengths grow at $2300\,\text{m}$ as depicted in Figure 8.26. Hence, a shrinking association ratio generally leads to increasing uncertainty, as we have fewer constraints on the vehicle location on the map.



(a) Number of seen facades along the trajectory.



(b) Ratio of associated points among all LiDAR measurements along the trajectory.

Figure 8.27: Evaluation of the association. The number of seen facades (Figure 8.27a) and the ratio of associated LiDAR points (Figure 8.27b) affect the localization accuracy.

8.3.1.3 All Datasets

After an in-depth evaluation of two exemplary trajectories, we want to evaluate the set-membership-based localization for all datasets. Therefore, we want to consider the average side lengths (long and short) of the smallest enclosing rotated rectangle and the average rotation interval width along the whole trajectories. Furthermore, to evaluate the rotation uncertainty, we depict the rotation interval width in the third column of Table 8.9. Table 8.10 summarizes the association performance where the first column depicts the number of facades and the second the average ratio of associated points.

We conducted two experiments with each T_1 and T_2 to study the effect of the map accuracy by using the LOD2 map and the OSM. The average rotated rectangle lengths and the rotation interval width are smaller using the LOD2 map since we account for a smaller facade error for the LOD2 map. Furthermore, due to the higher global consistency of the LOD2 map, our method can associate more points to facades on average compared to the OSM, as stated in the first two rows in Table 8.10. On average, our method uses one facade per frame more with the LOD2 map. Additionally, the ratio of associated points is almost $7\,\%$ higher for the LOD2 map.

Dataset	Average length of long rotated rectangle side	Average length of short rotated rectangle side	Average rotation interval width
T_1 with OSM	5.73 m	2.12 m	4.06°
T_1 with LOD2	3.56 m	1.57 m	2.16°
T_2 with OSM	4.59 m	2.36 m	3.66°
T_2 with LOD2	5.24 m	1.84 m	2.36°
0018	3.31 m	2.16 m	3.65°
0020	11.19 m	4.92 m	7.59°
0027*	5.04 m	2.5 m	4.08°
0027	3.25 m	2.3 m	3.42°
0028	10.78 m	5.82 m	5.04°
0033	6.88 m	4.64 m	4.56°
0034*	6.29 m	3.52 m	5.08°
0034	15.06 m	7.63 m	7.79°
0071	6.41 m	3.12 m	3.82°

Table 8.9: Evaluation of the set-membership results. The sizes of the position polygons are evaluated by determining the side lengths of the smallest rotated rectangle. We depict the average long-side and short-side lengths. The width of the orientation interval represents the rotation angle interval uncertainty.

In contrast to T_1, T_2 shows a different effect. As depicted in the third and fourth column of Table 8.9, only the short rotated rectangle side and the average rotation interval width are smaller for the LOD2 map compared to the OSM results – due to smaller uncertainties considered in LOD2. However, the rotated rectangle's large side is larger than the OSM results. Furthermore, the comparison reveals that the average number of seen facades and the ratio of associated points using the LOD2 map is slightly smaller than with the OSM according to Table 8.10. Hence, our method associates fewer measurements to the facades, due to which the long side of the rotated rectangle is larger with the LOD2 map.

Still, the question remains why fewer points are associated using the LOD2 map than OSM, although the LOD2 map is more accurate. Further investigation shows that the average values are mainly affected by the different levels of details of the maps. The OSM provides less accurate data and applies further simplifications on the building geometries. While the LOD2 map also considers smaller facade jumps between connected walls, the OSM typically approximates such structures by straight facades. At the beginning of the trajectory T_2, facades are observed by the LiDAR that are simplified in the OSM by straight lines. However, in the LOD2 map, the facade is represented by multiple connected walls that represent the complex geometry of the facade in more detail. Since we only admit associations with a minimal number of points associated with a wall, the association with the LOD2 suffers as many different walls are seen with fewer points associated per wall. In contrast, in the OSM, the considered facade is represented by one wall, and all points are associated with that wall. Hence, the number of associations per wall is naturally higher for OSM. Hence, using the LOD2 map, we miss those associations with smaller walls. As a result, the localization accuracy suffers in those situations using the LOD2 map. While this effect is less pronounced in T_1 and the later part of T_2, the impact of this effect leads to inferior average results in T_2 using the LOD2 map.

Dataset	Average number of seen facades	Average ratio of associated points
T_1 with OSM	2.3	27.7 %
T_1 with LOD2	3.37	34.4 %
T_2 with OSM	1.96	27.86 %
T_2 with LOD2	1.8	27.31 %
0018	3.27	12.21 %
0020	1.25	6.6 %
0027*	3.73	19.92 %
0027	4.38	19.1 %
0028	1.89	7.11 %
0033	1.41	4.46 %
0034*	2.09	6.43 %
0034	0.61	1.96 %
0071	2.05	6.47 %

Table 8.10: Evaluation of the LiDAR point and facade association. The table depicts the average number of seen facades and the average ratio of associated measurements.

The main problem we identified here is that our method assumes a fixed minimum number of points associated with the wall so that they are admitted as a valid set of associations. However, simply fixing this threshold as we do in our implementation is misleading since the number of points depends on factors such as the size of the wall and the distance. Hence, to improve our method, a dynamic threshold is more appropriate for the association.

For the KITTI trajectories, we only use the OSM for localization. According to Table 8.9, the localization uncertainty for 0018 and 0027 is the smallest. However, uncertainty for the localization estimate is significantly higher for 0020, 0028, and 0034 with a long rotated rectangle side above 10 m on average. The reason for the high uncertainty is a poor association of local LiDAR points to facades. The main problem in those trajectories is that vegetation, unmapped walls, and other objects occlude significant parts of the buildings. That is why the average ratio of associated points in Table 8.10 is significantly smaller. The refined localization module can only perform localization if points-to-facades association is possible.

8.3.2 Hybrid Approach vs. Maximum Likelihood Estimation

This subsection aims to compare our proposed bounded optimization within our hybrid refined localization approach with a classical MLE method that does not consider any bounds in the estimation. To ensure a fair comparison, we consider the same points-to-facades associations obtained by the best particles-based association procedure introduced in Part 6.2.2.1. In contrast to the previous subsection, where we evaluate the uncertainty of the set-membership results by considering the size of the determined set, we evaluate most likely pose estimates in this subsection. Consequently, instead of intervals, we have to deal with classical point-valued pose estimates. Accordingly, we use the classical error metrics for the evaluation: We determine the translation error by the Euclidean distance (root mean squared error) between the computed vehicle position and the ground truth position in the map for the considered time steps. As a

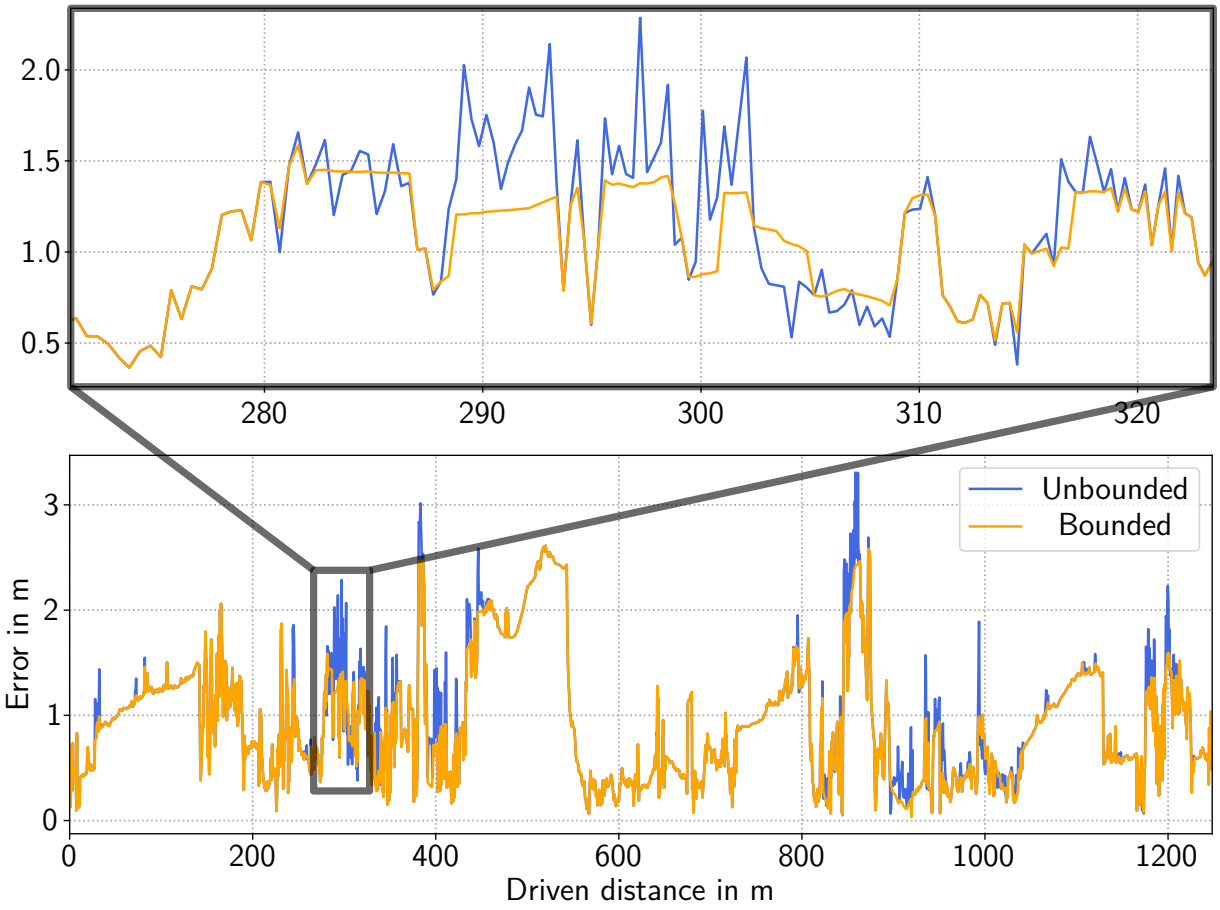


Figure 8.28: Root mean squared error of the MLE estimation in T_1 with the LOD2 map. Our bounded optimization is compared with the classical unbounded optimization.

simple scalar represents the rotation, we determine the error by the absolute error between the estimate and the ground truth. While we present in Part 8.3.2.3 the average errors for all datasets, we also provide in-depth evaluation for exemplary trajectories T_1 with the LOD2 map and 0027 with the OSM in Part 8.3.2.1 and Part 8.3.2.2, respectively.

8.3.2.1 Author-Collected Dataset T_1

Figure 8.30 shows an overview on the map and the trajectories for T_1 with the LOD2 map. The ground truth trajectory is colored green, the trajectory obtained by the classical unbounded optimization is blue, and the trajectory obtained by our bounded optimization approach is colored orange. As we can notice in the close-up figures, the bounded and unbounded optimization estimates are comparatively close to the green ground truth trajectory.

Since the minor deviations between the estimates are hard to recognize in the trajectory overview, we also provide the error plots for the translation in Figure 8.28 and for the rotation in Figure 8.29 along the trajectory. In both plots, the result of the bounded optimization is colored orange, and the unbounded results are colored blue.

In Figure 8.28, we can see that the bounded and unbounded optimization achieve similar results. However, our bounded optimization approach can reduce the error peaks. Especially the zoomed view on the error plot shows that the unbounded optimization leads to higher

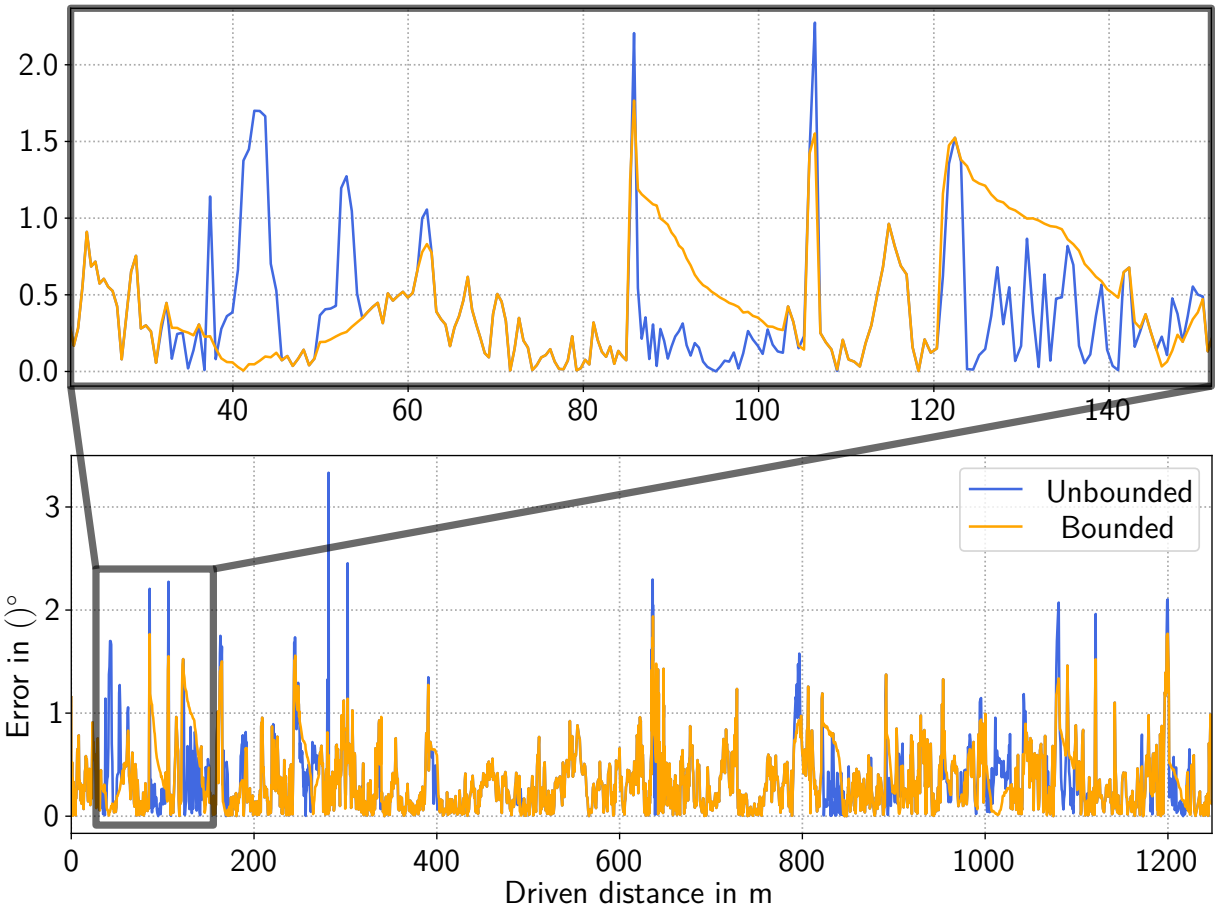


Figure 8.29: Rotation parameter error of the MLE estimation in T_1 with the LOD2 map. Our bounded optimization is compared with the classical unbounded optimization.

error, while in our approach, the bounds prevent the estimation from significant errors. This observation is also underlined by the overall results listed in Table 8.11. As depicted in the second row, the average translation and rotation errors are very similar. However, using the bounded optimization approach, the largest translation and rotation errors are significantly smaller.

While the bounds mainly improve the optimization results for the translation, the rotation error is shadowed by other effects: In Figure 8.29 in rare portions of the trajectory the blue curve is below the orange, which shows that the unbounded optimization sometimes leads to smaller errors – as it is the case in the close-up figure. This is because if an error-prone association leads to an incorrect contraction of the rotation interval, the interval may not contain the correct orientation angle. If this happens, the error will be higher for the bounded optimization as the bounds prevent the optimization from approaching the better solution. Since the uncertainty rises due to uncertainty accumulation of the odometry, the error of the bounded optimization is reduced slowly along the trajectory until the bounded and unbounded optimization provides the same error. This can be observed in the close-up in Figure 8.29 between $85\,\text{m}$ and $105\,\text{m}$ and between $120\,\text{m}$ and $150\,\text{m}$. This behavior of the bounded optimization is unfavorable. Luckily, this only happens in rare cases in practice since we choose the map uncertainty pessimistically.

Figure 8.30: T_1 trajectory overview and selected close-ups. The ground truth trajectory is green, our result is orange, and the unbounded optimization result is blue.

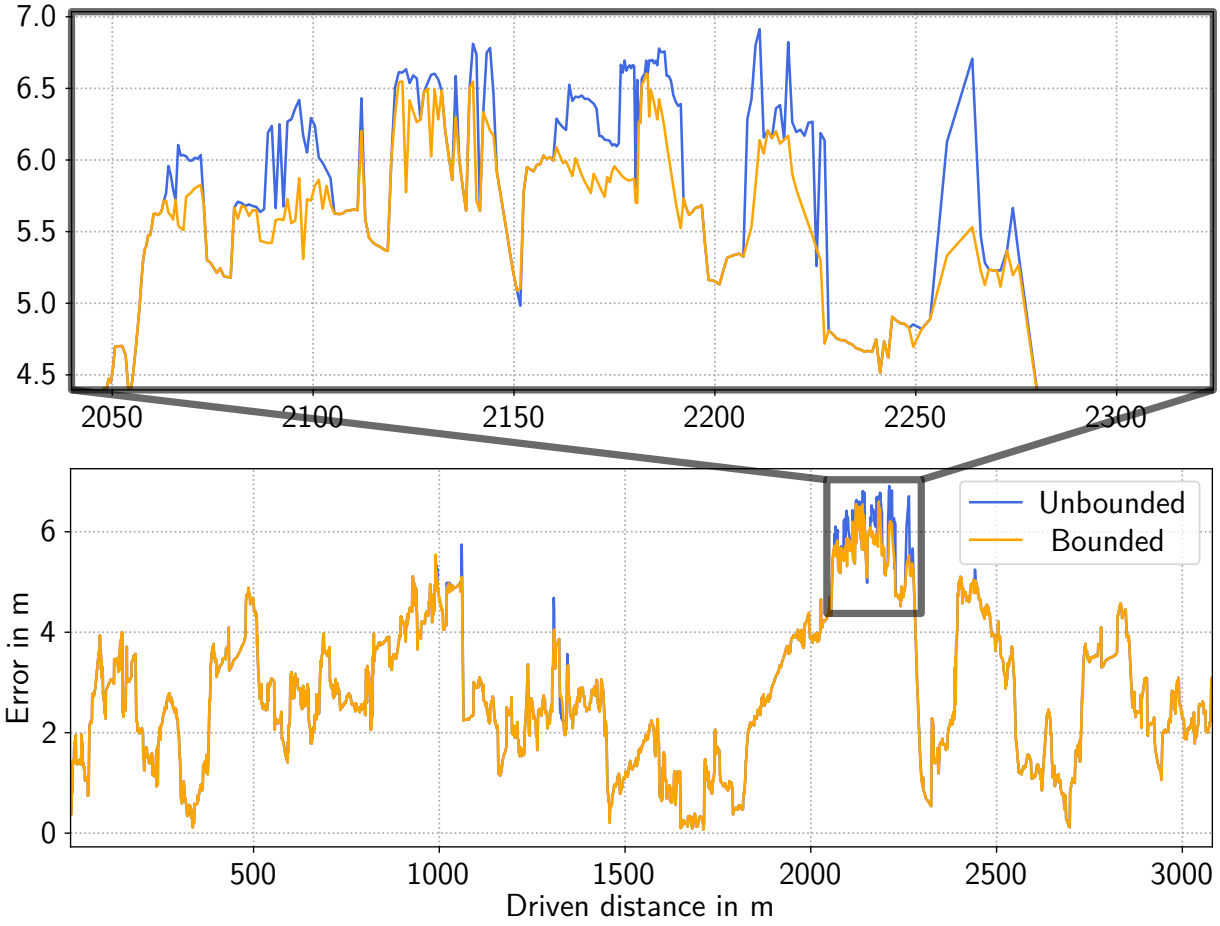


Figure 8.31: Root mean squared error of the MLE estimation in 0027 with the OSM. Our bounded optimization is compared with the classical unbounded optimization.

8.3.2.2 KITTI Dataset 0027

The overview and the close-up figures to the KITTI trajectory 0027 are depicted in Figure 8.33. While the optimization results in Figure 8.30 are very close to the green ground truth trajectory, in Figure 8.33, an offset between the estimated poses and the ground truth can be seen. Particularly in the top close-up figures, the offset is well observable.

If we consider the absolute Euclidean error of the translation in Figure 8.31, we can observe higher errors compared to Figure 8.28. Especially between $1700\,\text{m}$ and $2300\,\text{m}$, the error is above $6\,\text{m}$ for the bounded and unbounded estimate. However, we need to consider this error with a grain of salt since – as described above – the OSM and the ground truth trajectory have an unknown offset. Therefore, we assume that the actual error has to be lower in the range of the errors that we determined for T_1.

In comparing the bounded and the unbounded optimization, we can observe in Figure 8.31 and in Figure 8.32 only minimal differences. In general, also in this dataset, our method can reduce the error by preventing the optimization from divergence. This can again be observed in the close-ups in Figure 8.31 and Figure 8.32. While the unbounded optimization occasionally tends to have higher errors due to the divergence of the optimization process, our method can keep the error within a maximally permitted error.

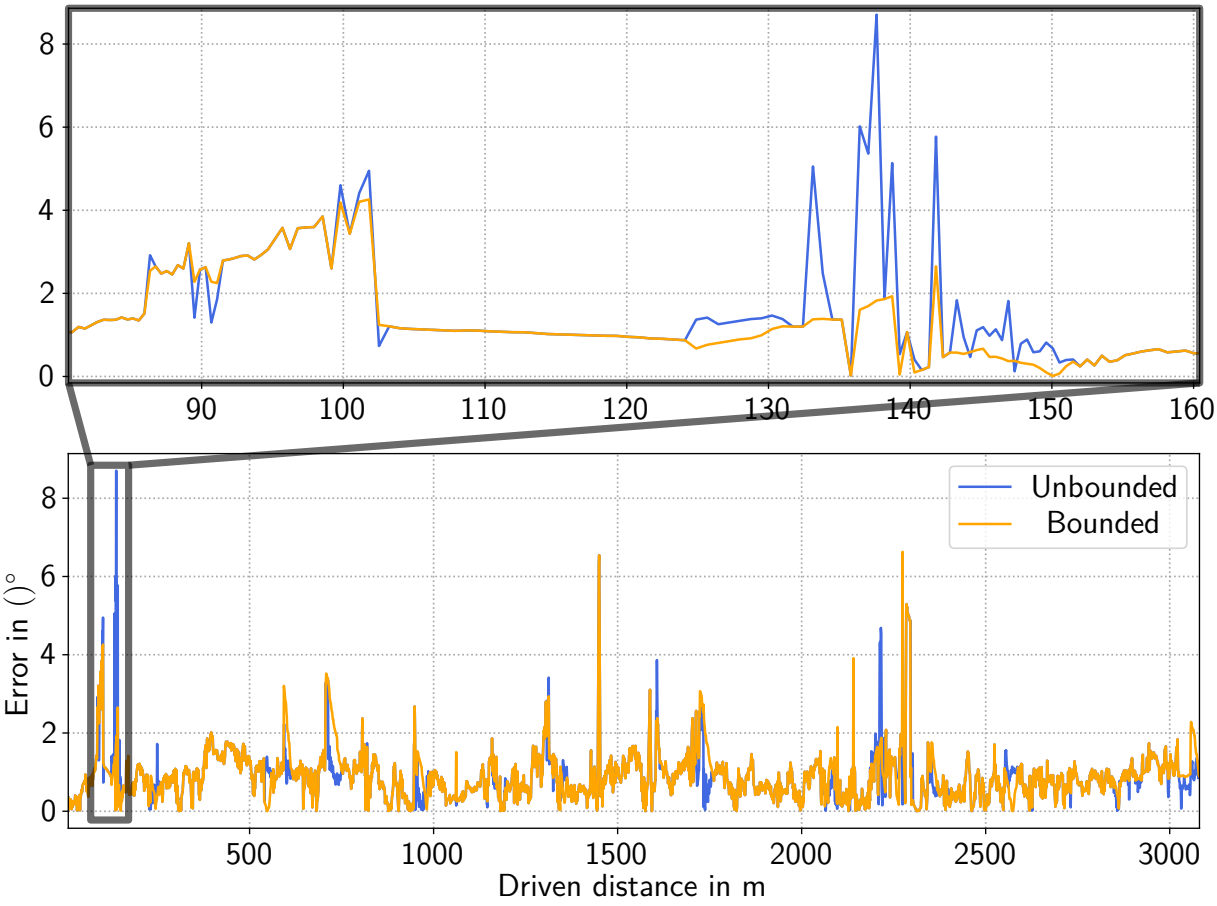


Figure 8.32: Rotation parameter error of the MLE estimation in 0027 with the OSM. Our bounded optimization is compared with the classical unbounded optimization.

8.3.2.3 All Datasets

The overall results for all datasets are listed in Table 8.11. The first two columns provide the average translation and rotation error for the whole datasets. The last two columns list the largest errors that are encountered in the datasets. Each column is further divided into two sub-columns – each provides the results for the bounded and unbounded optimization, respectively.

The first column shows that our proposed bounded optimization provides on average lower error for almost all datasets compared to the classical unbounded optimization. Nonetheless, the average difference between the translation errors is very small. Regarding the average rotation error, the differences between the unbounded and bounded optimized solutions only differ slightly. In contrast to the average translation error, the bounded optimization approach does not perform better than the unbounded version. The main reason for this observation is that the case of an incorrect association or overly optimistic local line extraction in the iHT leads to rotation intervals that provide bounds that hinder the optimization from converging to the minimal value, as explained above.

Nonetheless, the main advantage of using a bounded optimization is that the bounds prevent the optimization from significantly diverging. That is why we also consider the largest errors since minimizing this maximal error motivates the usage of bounded optimization. As presented

Figure 8.33: 0027 trajectory overview and selected close-ups. The ground truth trajectory is green, our result is orange, and the unbounded optimization result is blue.

Dataset	Average translation error		Average rotation error		Largest translation error		Largest rotation error	
	bound	unbound	bound	unbound	bound	unbound	bound	unbound
T_1 with OSM	**1.81** m	1.83 m	**0.68**°	0.7°	**5.51** m	6.2 m	**4.44**°	6.9°
T_1 with LOD2	**0.85** m	0.87 m	0.36°	**0.34**°	**2.61** m	3.31 m	**1.94**°	3.33°
T_2 with OSM	**1.89** m	1.9 m	0.99°	**0.94**°	**5.25** m	5.71 m	**6.04**°	11°
T_2 with LOD2	2 m	2 m	**0.72**°	0.75°	**5.27** m	5.89 m	**5.42**°	7.51°
0018	3 m	**2.99** m	**1.02**°	1.07°	**5.19** m	5.65 m	**3.36**°	5.62°
0020	**5.83** m	5.91 m	**1.4**°	1.42°	17.76 m	17.76 m	**3.97**°	6.99°
0027*	**2.26** m	2.27 m	0.63°	**0.62**°	**5.28** m	5.31 m	9.7°	9.7°
0027	**2.76** m	2.78 m	0.9°	**0.87**°	**6.6** m	6.91 m	**6.63**°	8.71°
0028	**5.3** m	5.32 m	**1.63**°	1.71°	**21.7** m	22.1 m	**12.05**°	15.65°
0033	**3.44** m	3.45 m	**1.13**°	1.25°	**6.39** m	6.45 m	9.26°	9.26°
0034*	**4.31** m	4.34 m	**1.06**°	1.1°	6.36 m	6.36 m	**4.04**°	11.59°
0034	**7.02** m	7.06 m	**3.83**°	4°	13.06 m	**13** m	**29.94**°	29.97°
0071	**3.05** m	3.09 m	1.59°	**1.41**°	**5.76** m	6.39 m	**5.41**°	9.41°

Table 8.11: Average error of the maximum likely pose. We compare a classical unbounded optimization with our bounded optimization. While the two left columns list the average translation and rotation error, the two right columns depict the largest errors.

in the last two columns of Table 8.11, the bounded optimization provides for almost every dataset a lower maximal error on average compared to the classical unbounded optimization. Our approach not only prevents the optimization from divergence but also provides information on the maximally possible error under consideration of the current association. Hence, it provides information on how trustworthy the estimate is with interval tools.

According to Table 8.11, the refined localization provides the best results with the T_1 dataset combined with the LOD2 map (second row). Here, we achieve an average localization error of below 1 m – simply using the surrounding building geometry. This shows the potential of the presented approach.

Using the OSM, our approach provides an average translation error of 1.81 m, which is acceptable regarding a map uncertainty of 1.5 m. However, since we restrict ourselves to buildings in the scope of this work as landmarks for localization, our method is doomed to fail in those datasets where we have poor visibility of the buildings. For instance, in the dataset 0034, large parts of the trajectory cover residential areas where the buildings are occluded by vegetation and unmapped walls. That is why this dataset's localization error is substantial.

In summary, the refined localization with the bounded optimization provides localization results with small errors depending on the map's accuracy. While our method provides a set of consistent positions as a polygon and a rotation interval describing the uncertainty of the location estimate, a bounded optimization approach provides the most likely point-valued pose considering the bounds of the consistent set of poses. This enables us to provide the maximum possible error for each frame, increasing the integrity of our localization approach.

Dataset	full	iHT	polygon	association
T_1 with OSM	0.025 s	0.02 s	0.000078 s	0.0036 s
T_1 with LOD2	0.033 s	0.028 s	0.00016 s	0.0042 s
T_2 with OSM	0.023 s	0.019 s	0.000063 s	0.0036 s
T_2 with LOD2	0.024 s	0.019 s	0.000063 s	0.0038 s
0018	0.012 s	0.01 s	0.000057 s	0.0028 s
0020	0.0071 s	0.0048 s	0.00003 s	0.0022 s
0027*	0.02 s	0.017 s	0.000059 s	0.003 s
0027	0.018 s	0.014 s	0.000064 s	0.0032 s
0028	0.01 s	0.007 s	0.000046 s	0.003 s
0033	0.009 s	0.006 s	0.0000325 s	0.0028 s
0034*	0.015 s	0.01 s	0.000046 s	0.0037 s
0034	0.011 s	0.007 s	0.000023 s	0.003 s
0071	0.014 s	0.011 s	0.000043 s	0.003 s

Table 8.12: Operation time evaluation for all datasets. All operation times per frame are averaged along the whole trajectories. The column *full* depicts the operation time per frame for the whole operation, *iHT* for the interval-based Hough Transformation, *polygon* for the polygon computation of the consistent set, and *association* for the particles-based association.

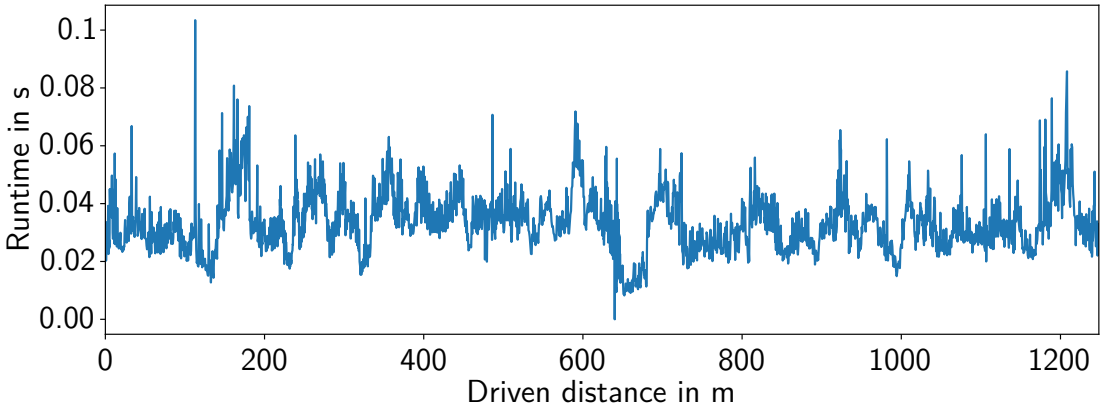


Figure 8.34: Full operation time per frame in T_1 with the LOD2 map along the trajectory.

8.3.3 Runtime

Conclusively we provide further insights into the runtime of the refined localization algorithm on our reference system. Figure 8.34 shows the operation times per frame along the trajectory for an exemplary run of T_1 with the LOD2 map. The overall average runtimes for the other trajectories are listed in the first column of Table 8.12.

Using the LOD2 map requires slightly more operation time per frame. The utilized LOD2 map contains a much larger region than the used OSMs. Hence, each map query takes slightly longer for the LOD2 maps than the smaller OSMs. While this runtime difference is noticeable for T_1, the runtime difference between the LOD2 and OSM-based run for T_2 is almost negligible. The reason for the observation is that in T_1, more facades are observed on average than in T_2 according to Table 8.10. As a result, in T_1, more map queries are performed so that the influence on the runtime in T_1 is higher.

Table 8.10 reveals that the main computational bottleneck of the hybrid refined localization approach is the iHT. While the iHT needs almost the major portion of the whole operation time per frame, the polygon estimation and the particle-based association have a minor contribution to the whole computation time. Since the iHT mainly depends on the number of points associated with facades in the map, the overall operation time per frame directly correlates with the associated points ratio. That is the reason why in Figure 8.34, there are runtime peaks at those driven distances where the association ratio (Figure 8.24b) rises.

On average, the complete refined localization algorithm needs $0.033\,\mathrm{s}$ per frame for T_1. As depicted in Table 8.12, the average runtimes for all other datasets are significantly below this value. Since we run our sensors with $10\,\mathrm{Hz}$, the refined localization is real-time capable.

8.4 HyPaSCoRe Localization System

In this section, we evaluate the whole HyPaSCoRe Localization pipeline using all datasets. Since the whole pipeline uses a particle filter that introduces a random component in the coarse localization, each run of the same dataset may provide different results. Hence, we consider multiple runs for each dataset to maintain a representative evaluation. While detailed explanations are given based on an exemplary run with the author-collected trajectory T_1 with OSM, the evaluation results for all datasets are condensed and summarized in tables. The values are determined by averaging the result for all 20 runs for each dataset.

Unique about our pipeline is that we distinguish between the localization estimate at localization time and in real-time. Since the windowed bundle adjustment delays the direct localization to real-time, we predict the real-time location based on the slightly delayed localization frame. Hence, this architecture raises the question of how the localization estimates differ regarding the accuracy and how significant the delay is. In Subsection 8.4.1, we provide a detailed evaluation of the impact of the localization results for the localization time frame and the real-time frame.

Another unique feature of our HyPaSCoRe Localization is that it provides a feasible and consistent set for the vehicle's location. While the coarse localization determines the feasible set, the refined localization computes the consistent set by performing maximum likely point-to-facade associations. In the scope of Subsection 8.4.2, we compare both sets regarding their pessimism and reliability, taking ground truth pose estimates into account. Furthermore, in Subsection 8.4.3, we analyze the integrity of the HyPaSCoRe Localization results by considering the Stanford-ESA integrity diagram.

In Subsection 8.4.4, the absolute localization errors of the maximum likely poses determined by the refined localization is evaluated. In particular, the difference between the localization results at localization time and real-time are compared. As mentioned above, the ground truth trajectory has centimeter accuracy consistent with the OSM and LOD2 map for the author-collected datasets. Unfortunately, the ground truth absolute pose provided by the KITTI dataset is unreliable. Although we evaluate the localization accuracy based on the faulty ground truth data, the error results for the KITTI datasets need to be considered with caution. In contrast, the results for the author-collected datasets are fully reliable.

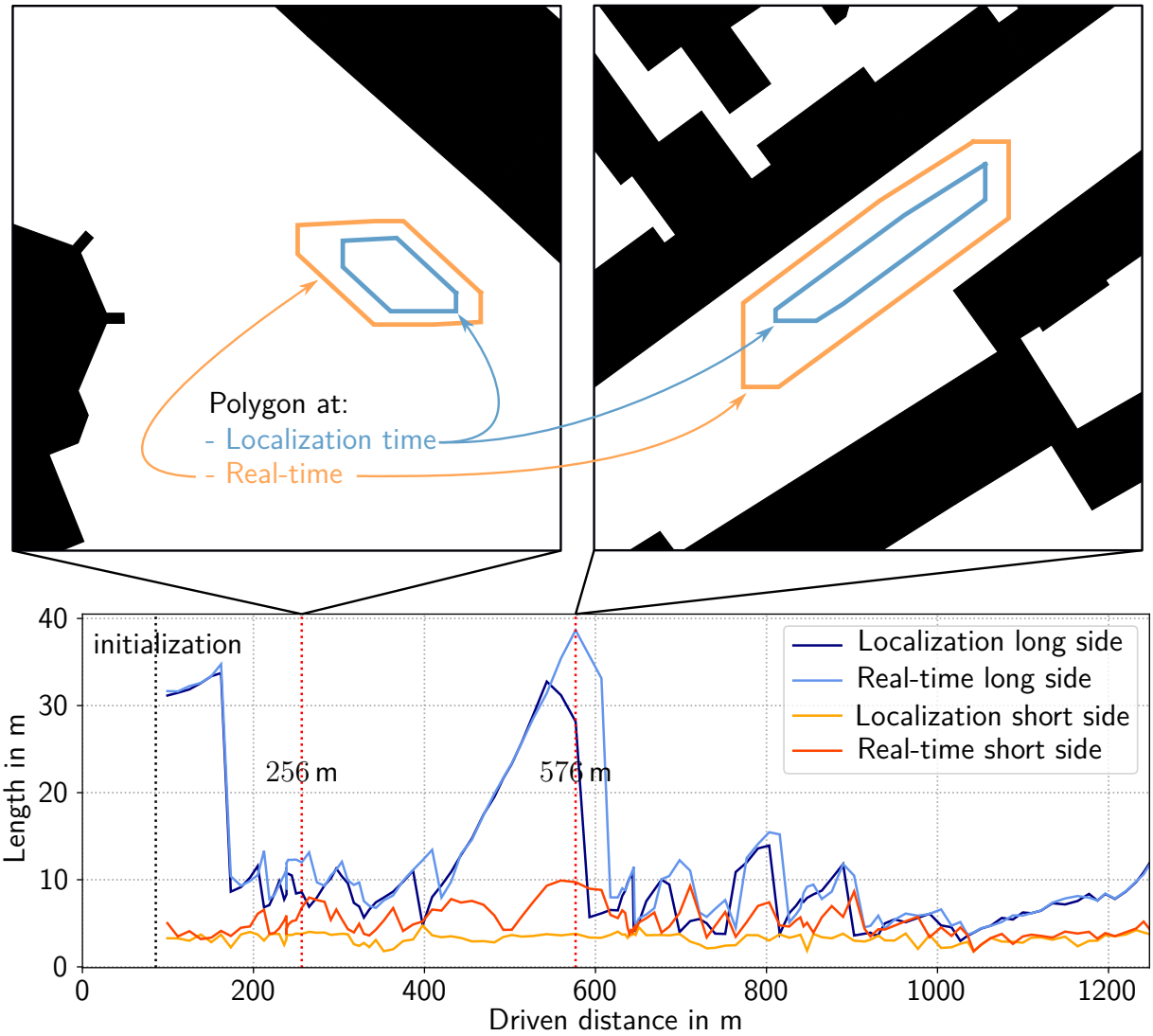


Figure 8.35: Comparison of the polygon sizes at localization time and real-time in T_1 using the OSM. The polygon sizes are measured by the long and short side lengths of the smallest rotated rectangles wrapped around the polygons.

We conclude the section with the runtime evaluation of all components of our HyPaSCoRe Localization in Subsection 8.4.5. The dedicated evaluation sections above evaluate the operation times for the individual components run independently. The main focus here is to provide insights into how well the different components simultaneously perform on one consumer-grade laptop.

8.4.1 Localization Time vs. Real-Time

The localization algorithms are applied to localization time and predict the real-time pose. The delayed execution of the localization algorithms is due to the windowed bundle adjustment that delays the localization times by the window time since only frames already processed by the windowed bundle adjustment are used for the localization.

In Figure 8.35, we compare the sizes of the polygons computed for the same frame at localization time and in real-time. Note that a frame determines a fixed pose in the map

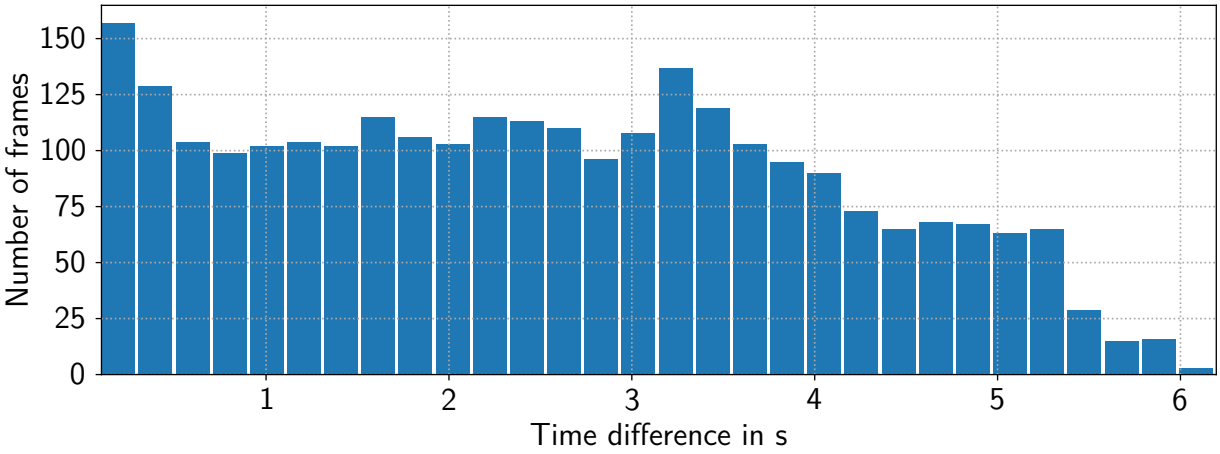


Figure 8.36: Histogram of the time delay between the localized and real-time frames along the trajectory T_1.

that describes the location of the vehicle at the time when the frame is acquired. Hence, due to the delayed localization time, we effectively estimate the pose of that frame twice: First, a real-time prediction (when the sensor provides the data) based on a previous localization frame processed in the localization pipeline. Second, the localization at localization time, when the localization pipeline reaches the considered frame when processing the frames after the windowed bundle adjustment sequentially. Generally, the polygon sides computed in real-time are larger than at localization time. Since we use the accumulated frame-to-frame odometry to determine the real-time pose based on the localization frame, the localization uncertainty of the frame, currently processed in the localization pipeline, is augmented by the frame-to-frame odometry uncertainty. Accordingly, it leads to higher uncertainties for the real-time estimates than the delayed estimate when the current frame is processed in the localization pipeline. We visually illustrate the different sizes of the position estimates for exemplary frames in the top images in Figure 8.35.

Note that the polygons are elongated along the driving direction in the right image. As already discussed in Subsection 8.3.1, this is due to the tube-like structure of the building that only constrains the location in the perpendicular direction. Consequently, as shown in the diagram, the short side of the rotated rectangle that encloses the polygon is small for that frame. The same behavior is also noticeable directly after the initialization of the refined localization.

Figure 8.36 shows the histogram of the time delays between the localization frame and the time at new sensor-data acquisition in T_1. The time delay depends on the window length, the time for the windowed bundle adjustment, and the time for the interval-based odometry computation. However, the most significant influence comes from the windowed bundle adjustment presented in Subsection 8.1.1 since good visual feature tracking can create windows spanning several seconds. The time delay mainly varies between $1\,\mathrm{s}$ and $5\,\mathrm{s}$. According to the histogram, there are more frames with a time delay of less than $1\,\mathrm{s}$ than with a delay larger than $5\,\mathrm{s}$. In general, larger delays imply larger visual tracking windows where more observations are considered in the windowed bundle adjustment, increasing the optimization's robustness. On the downside, a larger time delay between the localization frame and real-time frame also

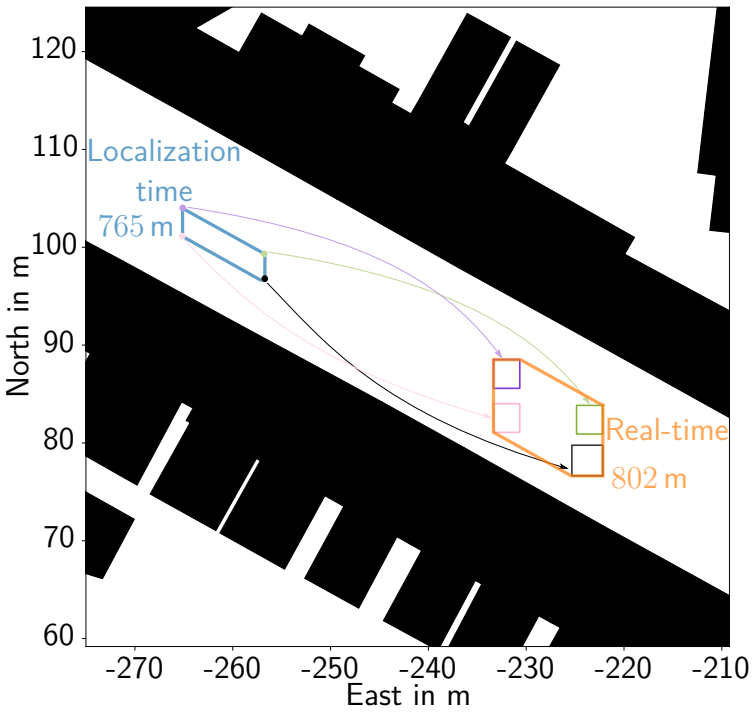


Figure 8.37: Exemplary polygons for the largest processing delay between the localization time and real-time. The blue polygon is determined for the frame that was localized. Since the localization is delayed, the real-time pose is predicted based on the visual odometry between the localization time and real-time. The uncertainty is accordingly inflated for the real-time pose estimate.

means higher uncertainty for the pose estimate in real-time since the prediction from the localization frame spans a more significant period in which more uncertainty is accumulated.

We illustrate the prediction of the position polygon in real-time for the largest time delay of 6.5 s. While the localization processes the frame at a driven distance of 765 m, the vehicle has already driven 37 m further, so the real-time frame is at a driven distance of 802 m. Due to the accumulation of the uncertainty of the frame-to-frame visual odometry as explained in Section 7.2, the real-time pose estimate has a larger uncertainty since the pose estimate at localization time is shifted and inflated to the real-time estimate.

The condensed average results of the localization time and real-time pose estimates of the refined localization for all datasets are given in Table 8.13. Note that the average results are determined based on 20 runs for each dataset. In Subsection 8.3.1, we already analyzed the results of the interval-based pose estimation of the refined localization. However, the experiments in Subsection 8.3.1 were conducted for the refined localization when executed independently. In contrast, we analyze the result in the whole HyPaSCoRe Localization here. This is also the reason why the results in Table 8.13 differ from those in Table 8.9: In Table 8.13, we only consider the polygon sizes after the initialization of the refined localization when the coarse localization provides stable particle-based estimates. In Table 8.9, the whole trajectory is considered since the localization is already restricted to a smaller area to initiate the refined localization for the experiment directly. Consequently, the results are not directly comparable.

Dataset	Average length of short rotated rectangle side		Average length of large rotated rectangle side		Average time delay
	Loc. time	Real-time	Loc. time	Real-time	
T_1 with OSM	3.6 m	4.8 m	9.2 m	10.3 m	2.5 s
T_1 with LOD2	3 m	4.1 m	6.6 m	7.5 m	2.6 s
T_2 with OSM	4 m	5.2 m	7.7 m	8.9 m	2.1 s
T_2 with LOD2	3.9 m	5.5 m	7.3 m	9.4 m	4.7 s
0018	3.1 m	4 m	4.2 m	5.5 m	0.6 s
0020	3.8 m	4.8 m	9.2 m	9.5 m	0.3 s
0027*	4 m	5.6 m	7.7 m	9.8 m	1.3 s
0027	3.3 m	4.3 m	4.6 m	5.7 m	0.5 s
0028	3.8 m	4.9 m	10.4 m	11.2 m	0.4 s
0033	10.8 m	11.4 m	24.4 m	25.6 m	0.47 s
0034*	4.2 m	5 m	5.8 m	7.8 m	1.9 s
0034	22.3 m	23.9 m	42.1 m	43.9 m	0.5 s
0071	3.6 m	4.5 m	21.1 m	23.2 m	1.5 s

Table 8.13: Size of the consistent set at localization time (loc. time) and in real-time for the same frame. Here we only consider the size of the polygon that we measure by the side lengths of the smallest rotated rectangle that can be wrapped around the polygon. We distinguish between the short and large sides to consider possible influences induced by the map structure. The last column lists the average time delay between the localization frame time and the real-time. The average values consider 20 runs for each dataset.

In Table 8.13, most datasets measure an average length of 3 m to 4 m for the short side at localization time while at real-time, the length is between 4 m and 6 m. Hence, according to our expectations, the real-time estimate is more uncertain than the localization time estimate. However, for the datasets 0033 and 0034, the polygons' sizes are significantly larger. The main issue is that facades are seen very rarely. Due to, for example, bushes, large front yards with tall trees, and walls, large parts of the facades are occluded. Hence, our building-map-based approach is not well suited for those datasets. Nonetheless, the HyPaSCoRe Localization pipeline can be adapted to other maps using the same methods presented in this work.

The large side of the polygons listed in the second and fourth columns show similar behavior as the short sides. However, the long side varies slightly more so that at localization time, we obtain lengths between 4 m and 10 m and at real-time between 5 m and 12 m. It is remarkable that for 0071, the large side has more significant uncertainty than the short side. This is due to the structure of the map. The whole trajectory goes through a tunnel-like structure with almost no constraints along the driving direction.

The uncertainty is comparatively large for all datasets, as we account for a wall uncertainty of 1.8m. We choose such a pessimistic uncertainty to tackle the map uncertainty, especially of the OSM, as some facades are incorrectly mapped. Although only a rare number of facades suffer from significant errors, our method also needs to consider those leading to more pessimistic uncertainty bounds.

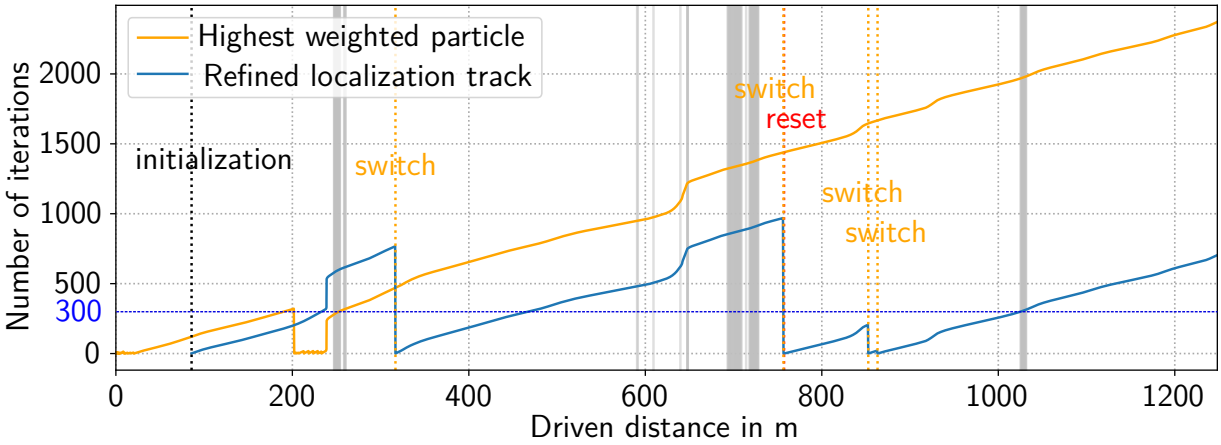


Figure 8.38: Highest particle age in the coarse localization and the refined localization tracking age. Orange and red dashed lines mark switches or resets of the track. The gray vertical lines define the driven distances at which the feasible set is contracted to the consistent set.

In the last column of Table 8.13, the average time delays between the processed localization frame time and real-time are listed. Interestingly, the author-collected datasets' average time delays are larger than for the KITTI datasets. In our investigations, we found that this is mainly due to the differing performance of the visual odometry module: Since in the author-collected datasets we use cameras that provide images with higher resolution, the optical tracking performs well and accordingly produces large tracking windows. Furthermore, a slightly larger average delay is noticeable for T_2 with the LOD2 map compared to T_1 and T_2 with OSM. One reason for this observation is that the coarse localization initially needs more processing time. Since the buildings are not densely distributed in that part of the map, the feasible set is not contracted as fast as in T_1. Additionally, the LOD2 map is significantly larger than the OSM counterpart, increasing the time for map queries. Consequently, multiple factors lead to different time delays.

8.4.2 Feasible Set vs. Consistent Set

Now we compare the feasible set $\mathcal{P}^{\boxplus}$ an the consistent set $\mathcal{P}^{\triangle}$. As explained in Section 7.3, we initialize the refined localization only if a stable particle of the bounded MCL in the coarse localization has survived a minimal number of iterations. We set the threshold for all datasets to 120 iteration, corresponding to $12\,\mathrm{s}$ of particle survival without being sampled out in the aggressive resampling procedure. Furthermore, after a track is initialized, the refined localization result can be used to contract the feasible set in the case of high reliability, the track can be reset, or a switch of the tracked particle in the refined localization can occur, as explained in Subsection 7.3.2. In this subsection, we will analyze when those events occur and the influences on the pose estimates. We will again base the detailed analysis on an exemplary run of T_1 with OSM.

Figure 8.38 depicts the particle age of the highest weighted particle in the coarse localization and the current track in the refined localization. As denoted in the diagram, the driven distance at which the refined localization is initialized is marked with a dashed black vertical line. Since

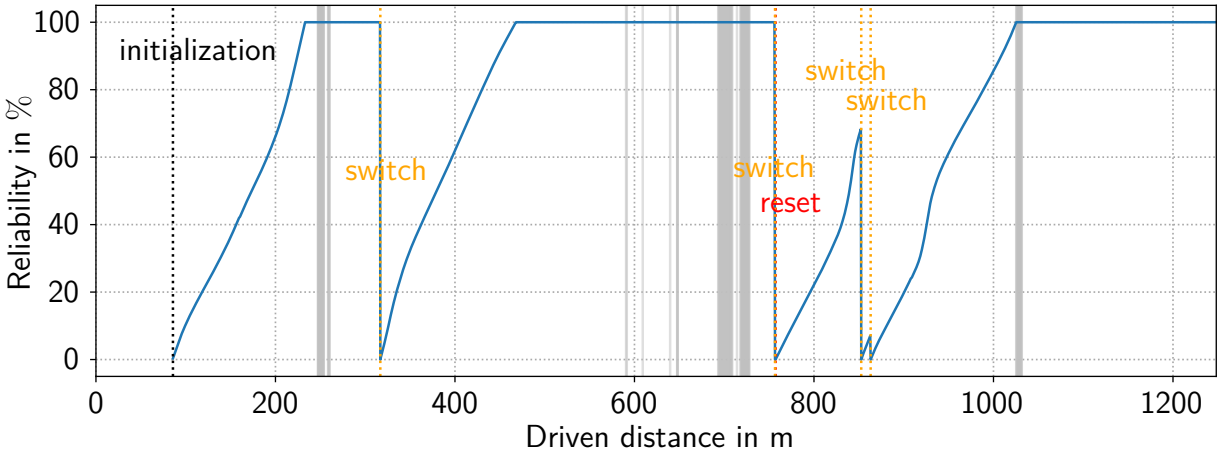


Figure 8.39: The reliability of the refined localization track is based on the age of the track. The contraction of the feasible set is only permitted if the reliability is $100\,\%$.

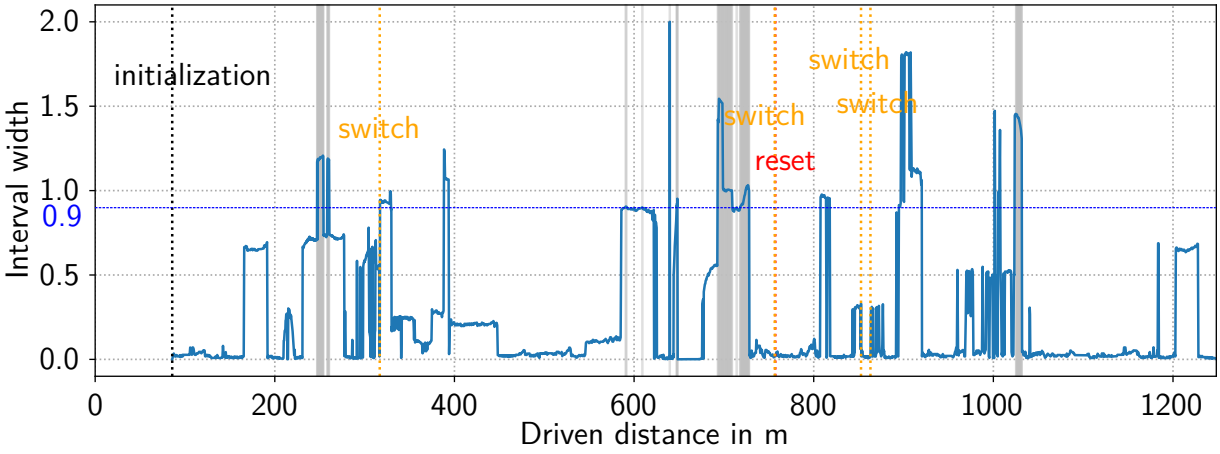
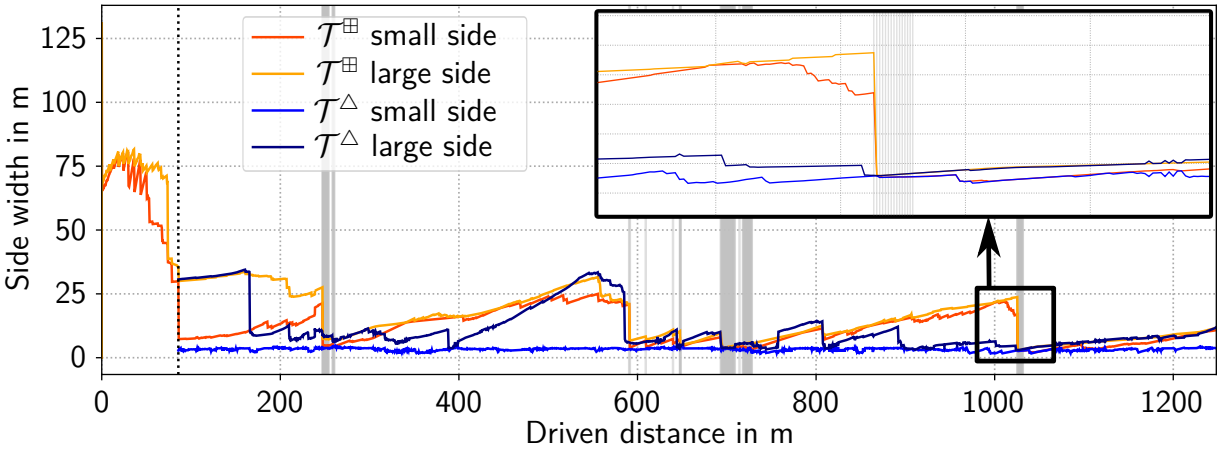


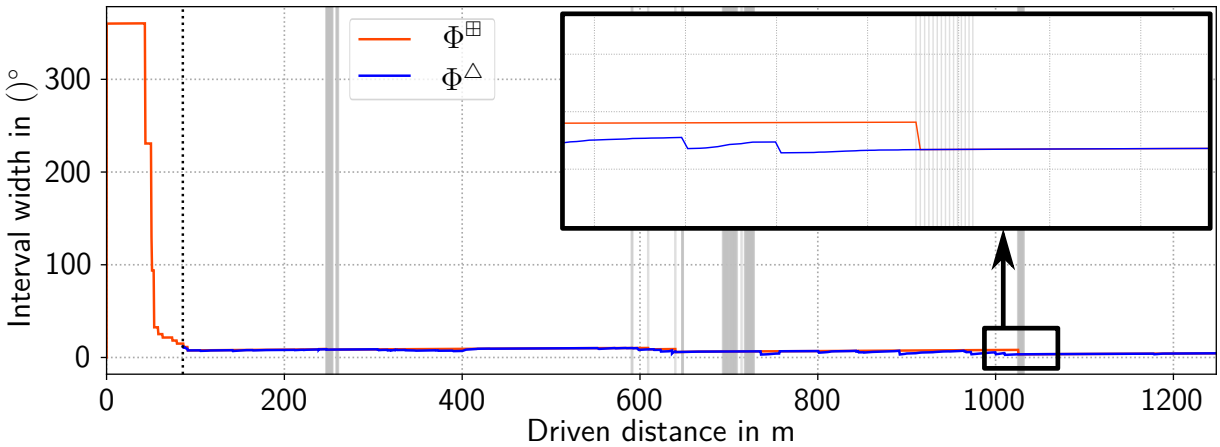
Figure 8.40: Interval width of the sine of the interval hull among all observed facade orientation angles. The contraction of the feasible set is only permitted for interval width above 0.9, which corresponds to almost $90°$ of minimal orientation difference.

the coarse localization already starts at the beginning of the trajectory, the yellow graph starts at $0\,\text{m}$ while the blue graph commences at $80\,\text{m}$. Note that the initialization corresponds to the driven distance at which the highest weighted particle reaches minimum iterations of survival above 120 (empirical thresholds). After the initialization, the highest-weighted particle and the refined localization track are treated independently. They are only compared regarding their weight which measures how well the pose estimates fit the current measurements to the map. Based on the weights, switches, and resets are performed. If those events occur (denoted by orange and red dashed lines), the age of the track is set to zero as explained in Section 7.3.

Note that the gray vertical lines that highlight those frames at which the feasible set is contracted only occur when the tracked particle age is above 300. This is because we determine a track's reliability only based on the age. Figure 8.39 shows the reliability of the track in the refined localization, which linearly scales the track age between $0\,\%$ and $100\,\%$. Hence, we fully trust the refined localization result if the age of the track is above 300 iterations, corresponding to a minimal tracking time of $30\,\text{s}$. This threshold is also selected empirically. Hence, as shown in Figure 8.39, the reliability is clipped to $100\,\%$ if the age exceeds 300 in Figure 8.38.



(a) Sizes of the rotated rectangle side lengths that enclose the feasible set $\mathcal{T}^{\boxplus}$ and the consistent set $\mathcal{T}^{\triangle}$. The side length measures the size of the sets.



(b) Interval widths of the feasible rotation angle $\Phi^{\boxplus}$ and consistent rotation angle $\Phi^{\triangle}$.

Figure 8.41: Size of the feasible set $\mathcal{P}^{\boxplus}$ and consistent set $\mathcal{P}^{\triangle}$ along the trajectory T_1. The feasible set is contracted to the consistent set for dedicated frames if the consistent set is staged as reliable.

As described in Subsection 7.3.2, the feasible set is only contracted if the consistent set is fully reliable and if differently oriented facades are observed. The difference in the facade orientation is determined by the interval width of the sine of the interval hull among all observed facades and is depicted in Figure 8.40. The sine interval angle width can take values between 0 and 2, while a 1 signals a facade orientation difference of $90°$. We set the minimum threshold to 0.9 so that if the sine interval angle width exceeds this value and if the consistent set is fully reliable, the feasible set $\mathcal{P}^{\boxplus}$ can be contracted to the consistent set $\mathcal{P}^{\triangle}$. The frames at which the feasible set is contracted are marked in all diagrams by a gray vertical line. The gray lines only occur if the reliability is at $100\,\%$ and the sine interval angle width is above 0.9.

The feasible and consistent set sizes are depicted in Figure 8.41. Here again, we use the rotated rectangle metric to measure the sizes of translation components $\mathcal{T}^{\boxplus}$ and $\mathcal{T}^{\triangle}$ and plot in Figure 8.41a the lengths of the small and short sides. Furthermore, the interval widths measure the orientation uncertainties and are plotted in Figure 8.41b. Note that if the feasible set contraction is triggered, the feasible set size is immediately contracted to the size of the

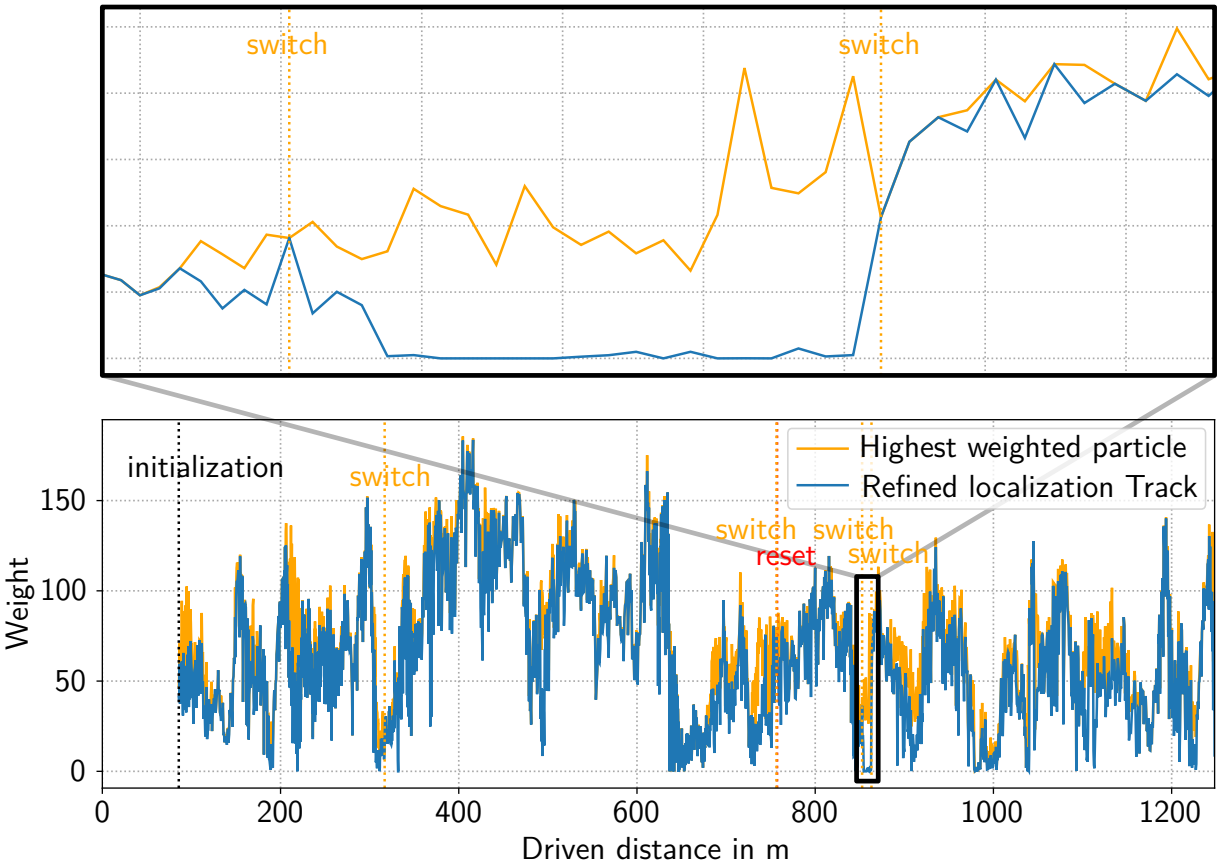


Figure 8.42: Weights of the highest weighted particle in the coarse localization and the weight of the tracked particle in the refined localization.

consistent set. Especially the close-up images in the top right of Figure 8.41a and Figure 8.41b visualize the contraction.

Switches and resets of the refined localization track are depicted in the diagrams by vertical dashed lines colored orange and red, respectively. Figure 8.42 exemplarily shows why we introduce those mechanisms into the HyPaSCoRe Localization pipeline. The rising uncertainty of the pose estimation due to the accumulation of the uncertain visual odometry may lead the refined localization track to drift. As a consequence, the weight of the track will decrease as the local measurements are not well matched to the map. For instance, this is the case in the exemplary run at $850\,\mathrm{m}$: As shown in the close-up image, the weight of the track is significantly lower than the highest-weighted particle. The reset and switch mechanism makes switching the track to the better solution possible. This is why the track has a higher weight after the second switch in the close-up. Nonetheless, we set the age again to zero so that the reliability, as shown in Figure 8.39, is set to zero again when switches or resets occur.

In Table 8.14, we summarize the comparison of the sizes of the feasible set and the consistent set for all datasets. Note that we only consider here the translation components. Table 8.15 summarizes for each dataset the driven distance until initialization, the number of feasible set contractions, switches, and resets. Furthermore, in the last two columns, we evaluate the ratio of frames whose feasible and consistent sets contain the ground truth, which measures the reliability of the computed sets.

Dataset	Average length of short rotated rectangle side		Average length of large rotated rectangle side	
	$\mathcal{T}^{\boxplus}$	$\mathcal{T}^{\triangle}$	$\mathcal{T}^{\boxplus}$	$\mathcal{T}^{\triangle}$
T_1 with OSM	11.9 m	3.6 m	16.9 m	9.2 m
T_1 with LOD2	6.9 m	3 m	9.3 m	6.6 m
T_2 with OSM	10 m	4 m	14.3 m	7.7 m
T_2 with LOD2	7.8 m	3.9 m	11.3 m	7.3 m
0018	13.3 m	3.1 m	16.7 m	4.2 m
0020	12.4 m	3.8 m	25.5 m	9.2 m
0027*	41.8 m	4 m	46.6 m	7.7 m
0027	13 m	3.3 m	15.8 m	4.6 m
0028	27 m	3.8 m	39.9 m	10.4 m
0033	33.8 m	10.8 m	46.1 m	24.38 m
0034*	33 m	4.2 m	39.8 m	5.8 m
0034	45.7 m	22.3 m	58.8 m	42.1 m
0071	22 m	3.6 m	38 m	21.1 m

Table 8.14: Comparison of the average sizes of the feasible set $\mathcal{T}^{\boxplus}$ and consistent set $\mathcal{T}^{\triangle}$. Only the translation components are considered here. We conducted 20 runs for all datasets to determine the average values.

As expected, according to Table 8.14, the consistent set is much smaller than the feasible set. The feasible set obtained by the coarse localization is only contracted by always valid and, therefore, elementary constraints. In contrast, the consistent set considers the local facade matching of the LiDAR points, which provides smaller but less reliable results. Although we contract the feasible set in the case of high reliability of the consistent set, the contractions only occur rarely – and for some datasets, never (cf. Table 8.15). This is also why the reliability of the feasible set is higher than that of the consistent set, as we will discuss later based on Table 8.15.

Comparing the sizes of the sets among all datasets, the feasible set size varies more than the consistent set size. Recall that the No-Overlap Contractor and the No-Cross Contractor contract the feasible set. Hence, the map's topology and the buildings' density significantly influence the size of the feasible set. As we have chosen the datasets in such a way that we cover and evaluate the HyPaSCoRe Localization based on different scenarios, this leads to more considerable differences. The consistent set size mainly depends on the visibility and matching performance of the local LiDAR data to the map facades. For instance, in 0034 and 0033, facades are barely observed. That is why the consistent set size is so large for those datasets.

Another interesting observation is that 0027* measures comparatively small consistent sets, although the feasible set is very large. In Table 8.15, we can find the reason: The consistent set is never reliable enough, so the feasible set is never contracted to the consistent set. Apart from a few exceptions, the consistent set size seems to vary less across the datasets since we assume similar facade uncertainties.

In the first column of Table 8.15, the driven distances until initialization are listed. The driven distances differ significantly from dataset to dataset since the initialization heavily

Dataset	Average Driven distance initial.	Median Num. of exploration region contraction	Median Num. of switches	Median Num. of resets	Average Reliability feasible set	Average Reliability consistent set
T_1 with OSM	81.4 m	88	8.5	3	100 %	85 %
T_1 with LOD2	180 m	245	1	0	100 %	99 %
T_2 with OSM	149.6 m	336	8	3	100 %	92 %
T_2 with LOD2	88.3 m	170.5	12	10	100 %	96 %
0018	195 m	196	8	7	(83 %)	(46 %)
0020	425.4 m	0	5	4	(73 %)	(36 %)
0027*	216.3 m	0	3	1	(96 %)	(71 %)
0027	259.7 m	733.5	7.5	2.5	(73 %)	(32 %)
0028	321.1 m	104	34	16	(92 %)	(38 %)
0033	245.3 m	0	12	7	(100 %)	(58 %)
0034*	747.8 m	0	0	0	(93 %)	(7 %)
0034	619.2 m	0	29	14.5	(100 %)	(58 %)
0071	165.4 m	0	3	0	(100 %)	(68 %)

Table 8.15: Evaluation of various aspects of the feasible set and the consistent set. The first column lists the average driven distance until the refined localization is initialized. The second column depicts the median number of contractions of the feasible set to the consistent set, the third column the number of switches, and the fourth the number of resets. The last two columns depict the reliability of the feasible and consistent set. The reliability is measured based on the ratio of frames at which the respective set encloses the ground truth.

depends on the map structure. Therefore, the results are not directly comparable among the datasets. Nonetheless, the initialization driven distance is a disadvantage of the HyPaSCoRe Localization: The vehicle has to drive a significant distance until the refined localization can operate. Hence, the localization approach does not provide appropriate localization results directly from the starting point onward. The user has to drive the vehicle initially to reduce the localization uncertainty.

In the second column, it is noticeable that the feasible set contraction to the consistent set is not performed at all for some datasets because the consistent set is not reliable enough. That is also why the feasible set size for those datasets is large. The feasible set is not contracted for almost half of the datasets, leading to high pessimism. However, pessimism is necessary because false contractions of the feasible set will corrupt it so that it does not contain the correct pose anymore, which we seek to avoid as we want to obtain a reliable feasible set with high integrity. The number of switches and resets differ in each run as they depend on the performance of the bounded MCL that introduces random resampling.

The last two columns depict the reliability of the feasible and consistent set. We measure the reliability by computing the ratio of frames whose feasible and consistent sets contain the ground truth pose. The ground truth pose is always inside the feasible set for the author-collected datasets. Consequently, the feasible set is entirely reliable. In contrast, the consistent set has reliability below $100\,\%$, which indicates that incorrect facade matches occasionally corrupt

the consistent set. Regarding the KITTI dataset, the reliability analysis in Table 8.15 is not trustworthy because we encounter a significant offset between the ground truth poses provided by the publishers of the KITTI datasets and the OSM data. That is why the KITTI results are in brackets and only listed here for completeness.

In summary, the feasible set has high integrity. The correct pose is always included in the set. However, the feasible set can be very pessimistic. The consistent set, which is less pessimistic and provides tighter bounds, has lower integrity standards. Nonetheless, in the case of high reliability of the consistent set, we can reduce the pessimism of the feasible set and gain tighter bounds without losing the high integrity standard.

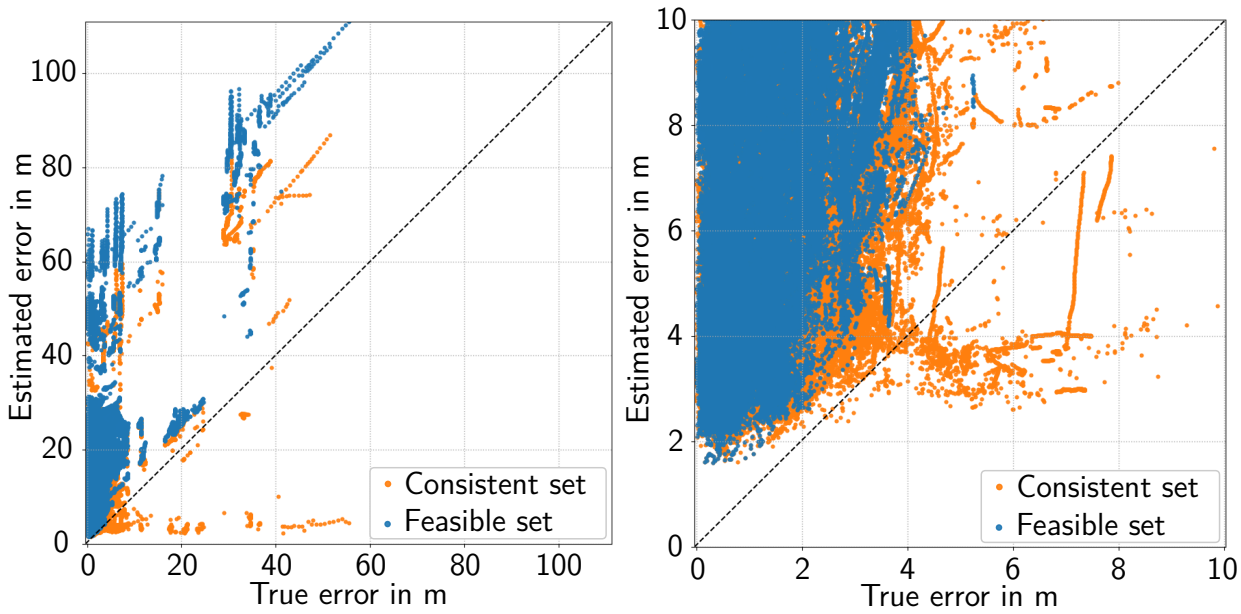
8.4.3 Integrity Evaluation

To assess the integrity of the HyPaSCoRe Localization, we analyze the Stanford-ESA integrity diagram introduced in Chapter 1. The Standford-ESA diagram plots the true and estimated errors as tuples for each localized frame (cf. Figure 1.1). While points in the top left triangle ensure the safe operation of the system, points in the bottom right lead to dangerous underestimation of the error.

To evaluate the integrity of the HyPaSCoRe Localization, we only consider the author-collected datasets T_1 and T_2 since we only have reliable ground truth for those trajectories. In our evaluations, we conducted 20 experiments for each dataset. As we have the OSM and LOD2 map for the T_1 and T_2, we consider 80 runs in the integrity assessment. Here, we only consider the translation part and neglect the rotation component for the simplicity of the Stanford-ESA integrity diagram.

In our experiments, the true error is determined by the error between the most likely estimate of the refined localization and the ground truth for each frame. The feasible and consistent set size determines the uncertainty of the most likely solution at each frame. However, to compute the estimated error for a frame, we exploit the rotated enclosing rectangle approximation of the sets: For each frame, we compute the Euclidean distance of each corner of the rotated rectangle to the most likely position. We define the largest distance as the estimated error. Each frame within each run provides a tuple of the true error and the estimated error that we plot for all 80 runs in the Stanford-ESA diagram depicted in Figure 8.43.

The diagram shows that the feasible set never underestimates the error and provides reliable uncertainty estimates. In contrast, the consistent set occasionally underestimates the error, as we can also see orange points on the bottom right triangle. The feasible set satisfies high integrity standards as it never leads to misleading information. The contraction of the feasible set to the consistent set reduces the pessimism while not reducing the reliability of the HyPaSCoRe Localization estimates. This again underlines our findings in the previous subsection 8.4.2. Nonetheless, the uncertainty is sometimes vastly overestimated by the feasible set. Note that we have comparatively large uncertainties because only building maps are used. Including further maps with other landmarks can reduce the pessimism.



(a) Full view.

(b) Zoomed view.

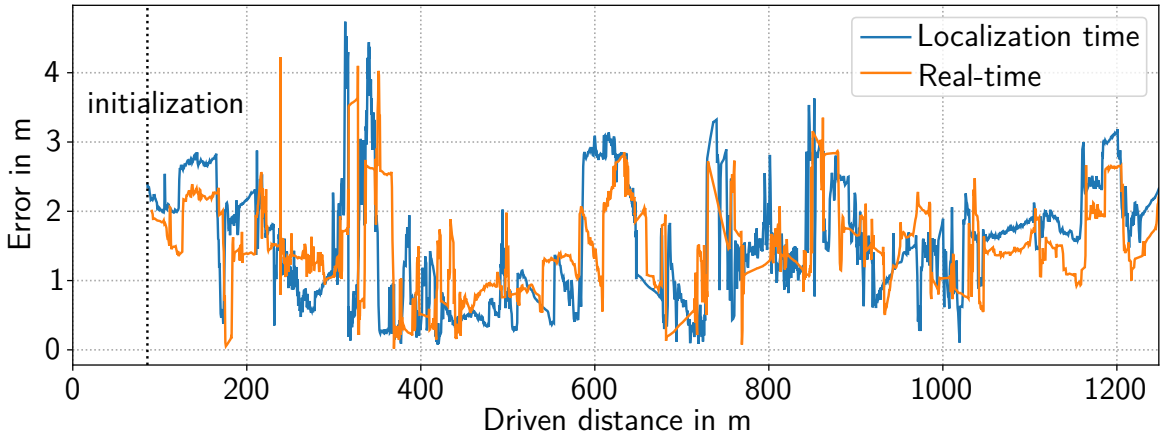
Figure 8.43: Stanford-ESA integrity diagram for the HyPaSCoRe Localization system. In this diagram, only the author-collected datasets, including LOD2 and OSM runs, are considered. For each dataset and map, 20 experiments are conducted to consider each frame of 80 runs in the integrity diagram. In this diagram, only the translation part is considered. Figure 8.43a depicts the complete diagram while Figure 8.43b shows a close-up of the critical part.

8.4.4 Localization Error

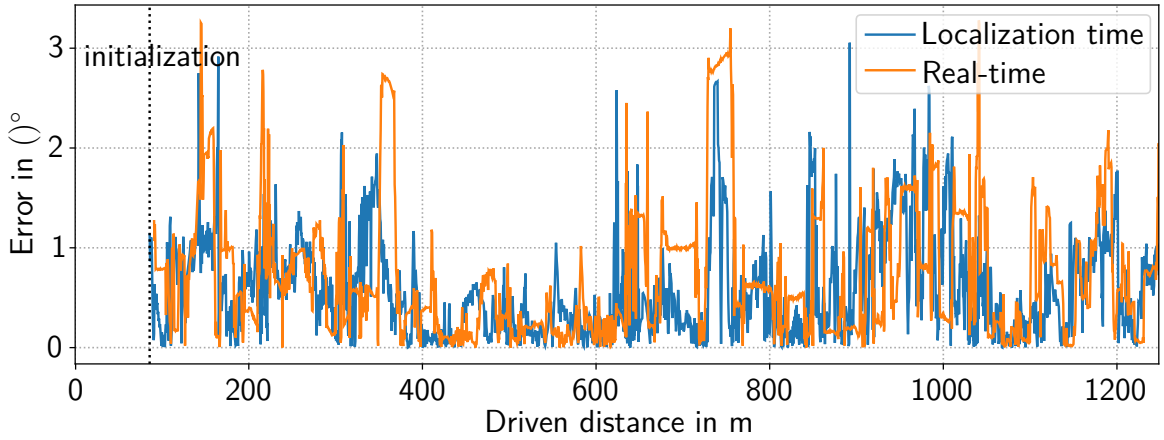
Now we consider the absolute error of the most likely poses determined by the refined localization. Therefore, we consider the ground truth poses of the datasets for each frame. Note that we only determine the error after the initialization of the refined localization. Remember that while the ground truth absolute poses in the map frame are reliable for the author-collected datasets, the ground truth data of the KITTI datasets need to be considered with caution as offsets between the ground truth and the OSM exist. Therefore, the ground truth trajectories for the KITTI datasets are only considered as approximations.

Figure 8.44 shows the absolute localization error at localization time and in real-time for the whole trajectory of T_1 with the OSM. We plot the real-time, localization time, and ground truth trajectories in the map overview in Figure 8.45 to provide a more intuitive visualization of the error. While the absolute translation error in Figure 8.44a reaches up to $4\,\mathrm{m}$, during the major part of the trajectory, the error stays below $2\,\mathrm{m}$. The absolute rotation error depicted in Figure 8.44b has rare peaks up to $3°$ while for the major part of the trajectory, the error is below $1.5°$.

According to Figure 8.44, the translation and rotation errors are slightly larger for the real-time estimates. Table 8.16 underlines the observation. However, the small difference in the error is almost negligible. The curve structure at localization time and in real-time is similar along the trajectory. However, the real-time error curve is slightly delayed. This is because the error made at localization time is propagated to the real-time pose estimation.



(a) Translation error.



(b) Rotation error.

Figure 8.44: Absolute pose estimation error at localization and real-time.

In Figure 8.44a, a longer series of larger errors (at almost $3\,\mathrm{m}$) is noticeable at a driven distance of $600\,\mathrm{m}$ to $700\,\mathrm{m}$. The close-up of this part of the trajectory is shown in the left middle close-up in Figure 8.45. Our investigations reveal that in this part of the trajectory, the bottom left building facades have a larger positioning error as they are shifted in the north direction. Since the facade is still matched with the local LiDAR data, the trajectory is slightly shifted to the north leading to the observed larger errors.

Furthermore, between $700\,\mathrm{m}$ and $800\,\mathrm{m}$, we can observe error jumps. The corresponding close-up figure is in the bottom left of Figure 8.45. Here we also found that the series of buildings on the south side (bottom part in the close-up) are slightly shifted to the north leading to more significant errors. However, since the width between the north and south buildings are not consistent anymore due to the offset, in our experiments, we experience that the localization estimate jumps since either the left or the right facade is matched. That is why the blue trajectory in the close-up has spikes along this part. Hence, this is a disadvantage of the HyPaSCoRe Localization since constantly checking for better poses in the feasible set tends to cause the localization estimate to jump. Since this improves the convergence behavior of the localization, the tracking behavior suffers.

Although our method can cope with the uncertainty of the OSM by enclosing the correct pose in the feasible set, the most likely estimate suffers from inconsistent maps. Hence, to

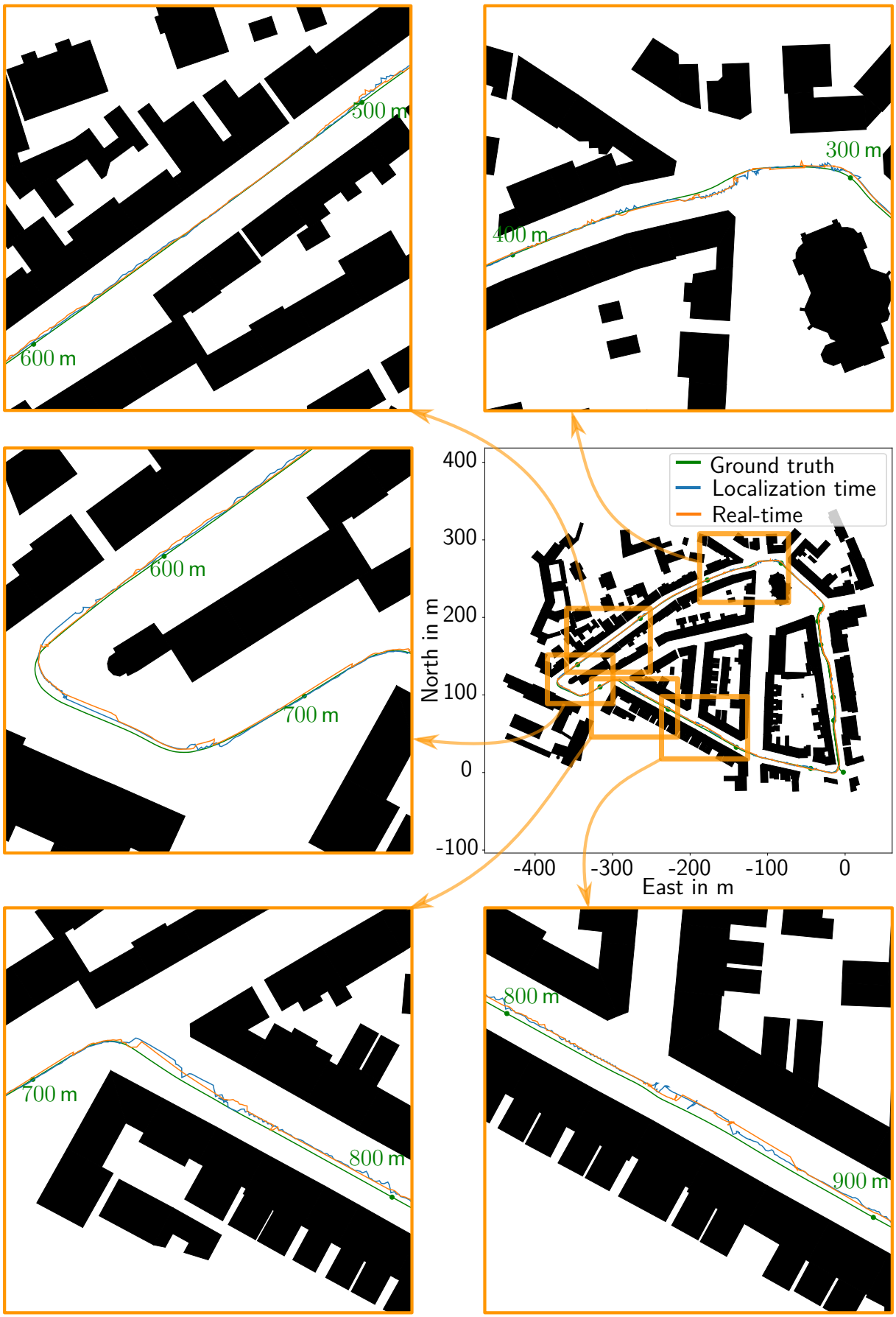


Figure 8.45: Trajectories plotted into the map with close-ups for T_1 with OSM.

Dataset	Average translation error		Average rotation error	
	Loc. time	Real-time	Loc. time	Real-time
T_1 with OSM	1.72 m	1.72 m	0.54°	0.62°
T_1 with LOD2	0.82 m	0.86 m	0.23°	0.36°
T_2 with OSM	1.33 m	1.46 m	0.77°	0.8°
T_2 with LOD2	1.03 m	1.13 m	0.56°	0.66°
0018	4.22 m	3.93 m	1.73°	1.46°
0020	4.95 m	5.66 m	1°	1.1°
0027*	3.28 m	3.67 m	1.01°	1.04°
0027	3.22 m	3.58 m	0.82°	0.98°
0028	5.06 m	5.53 m	1.14°	1.28°
0033	9.1 m	9.4 m	1.75°	1.93°
0034*	7.4 m	7.8 m	1.31°	1.43°
0034	13.5 m	14 m	2.52°	2.59°
0071	3.5 m	4.1 m	2°	2.1°

Table 8.16: Absolute average localization errors at localization time and in real-time. The average results consider all 20 runs per dataset.

obtain low localization errors, it is vital to use consistent and accurate building maps such as the LOD2. In that case, the absolute localization error can be below 1 m according to Table 8.16.

Table 8.16 lists all datasets' average absolute localization errors. The results consider all 20 runs for each dataset. Generally, comparing the errors at localization time and real-time reveals that the error is slightly larger at real-time, but the differences are insignificant. The absolute translation and orientation errors are significantly smaller for the author-collected datasets T_1 and T_2. The HyPaSCoRe Localization shows its best performance on T_1 with the LOD2 map. Since T_1 covers a region densely populated by buildings, and the LOD2 map provides accurate facade locations, our building map-based localization approach provides estimates with an average error below and close to 1 m, which is very good for a building map only localization.

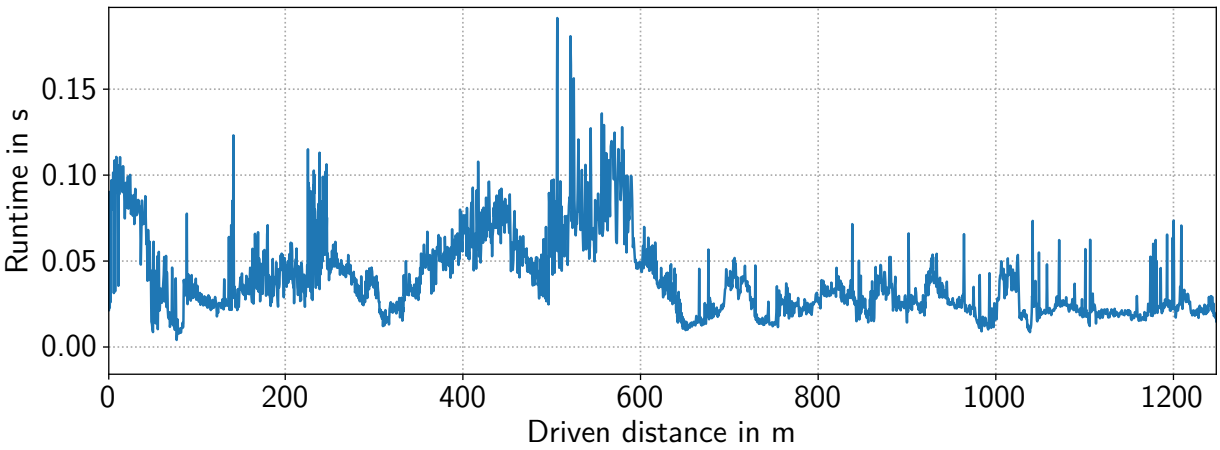


Figure 8.46: Operation times per frame for the localization thread plotted along the trajectory T_1 with the OSM.

Dataset	Visual odometry	Coarse localization	Refined localization	Full localization
T_1 with OSM	0.03 s	0.008 s	0.025 s	0.036 s
T_1 with LOD2	0.029 s	0.007 s	0.026 s	0.035 s
T_2 with OSM	0.03 s	0.005 s	0.017 s	0.028 s
T_2 with LOD2	0.03 s	0.006 s	0.02 s	0.034 s
0018	0.03 s	0.01 s	0.01 s	0.027 s
0020	0.03 s	0.008 s	0.006 s	0.016 s
0027^*	0.03 s	0.03 s	0.02 s	0.058 s
0027	0.03 s	0.013 s	0.019 s	0.033 s
0028	0.029 s	0.01 s	0.011 s	0.024 s
0033	0.031 s	0.016 s	0.011 s	0.03 s
0034^*	0.031 s	0.017 s	0.004 s	0.025 s
0034	0.034 s	0.037 s	0.01 s	0.052 s
0071	0.036 s	0.01 s	0.004 s	0.021 s

Table 8.17: Average operation times per frame among all 20 runs per dataset. The first column only covers the front-end of the visual odometry. The runtime of the particular localization methods and the full localization are listed in the first, second, and third columns, respectively.

The orientation error is very low as the long facades constrain the orientation well. However, according to our expectations, the same trajectories combined with the less accurate OSM data provide higher average errors, which are still significantly below 2 m for the translation part.

In contrast to the author-collected trajectories, the KITTI datasets have significantly larger errors. While in 0027, the average error is at 3.22 m at localization time, 0034 shows the highest error with 13.5 m, which is almost unusable. On the one hand, the higher errors mainly come from significant offsets of the OSM data to the ground truth. Furthermore, in 0034, facades are barely seen, so the building map-based localization does not work well. While the error results of the author-collected datasets are entirely trustworthy, the results of the KITTI datasets need to be considered with caution due to the ground truth problem.

8.4.5 Runtime

Conclusively we evaluate the whole HyPaSCoRe Localization system's runtime when all modules run on one machine. Therefore, we must consider the parallel processing architecture introduced in Section 7.1. In total, we have three threads that run in parallel. One thread performs real-time sensor data processing and visual feature tracking as the front-end in the visual odometry module. Another thread in the visual odometry module performs the windowed bundle adjustment. The third thread applies the coarse and refined localization algorithms.

Since the windowed bundle adjustment is processed rapidly, the only bottlenecks for real-time operation are the front-end processing in the visual odometry and the localization algorithms. That is why in Table 8.17, we only consider the average runtime per frame for those components. The operation times for the entire localization thread per frame are plotted for T_1 in Figure 8.46. Besides the peaks, the localization time is significantly lower than the real-time operation limit of 0.1 s.

During our experiments, we noticed that the initial contraction of the feasible set strongly influences the performance. The coarse localization becomes slow when the feasible set contains many subsets within the subpaving. Mainly for maps that are sparsely populated with buildings, this poses a performance problem. While the front-end performs very similarly on all datasets, we can see some differences in the performance of the localization algorithms. For instance, the operation time for the whole localization thread is significantly larger for 0027^* and 0034 because the feasible set is not contracted small enough as in the other datasets. Hence, the time for the coarse localization increases. Furthermore, for those datasets that do not contract the feasible set since the consistent set is never staged as reliable enough according to Table 8.15, we can observe larger operation times in Table 8.17 regarding the coarse localization and thus for the full localization thread. Nevertheless, the operations times still enable real-time operation of our proposed HyPaSCoRe Localization.

9 Summarizing Discussion and Prospects

The experimental evaluation aimed to validate the HyPaSCoRe Localization and to reveal the benefits and problems of the hybrid approach compared to State of the Art methods. Therefore we validated each component independently and compared the results with other benchmark approaches. Finally, we evaluated the whole HyPaSCoRe Localization system under consideration of the accuracy, reliability, and real-time operation. In the following, we sum up the results and lessons learned. Furthermore, we provide hints and ideas for future works to improve our HyPaSCoRe Localization.

Visual Odometry

In Section 8.1, we evaluated the visual odometry representing the core module for the relative motion estimation. The module is divided into two parts: front-end and back-end. We showed that the windowed bundle adjustment combined with a two-staged robust kernel optimization makes detecting outliers with high reprojection errors possible. In contrast to State of the Art approaches, we do not only discard single observations that lead to high residuals. We discard all observations of landmarks that are connected to at least one outlier observation. Although this outlier rejection mechanism is conservative, it effectively deletes incorrect landmark observations. Furthermore, we showed that the remaining residuals can be used to assess the observation error on the image. By doing this, we avoid conservative uncertainty assumptions on the pixel error. On the one hand, this design heavily relies on the correct convergence of the windowed bundle adjustment, which we cannot guarantee. It represents a critical point in the visual odometry. On the other hand, we also cannot guarantee that all observations meet an assumed conservative pixel error. Furthermore, in the case of the windowed bundle adjustment divergence, the pixel observations' uncertainty will be large as the residuals will become large, or the interval-based odometry estimation will provide empty sets, which directly signals that the observations are inconsistent. Nonetheless, in practice, the experimental results showed that our uncertainty assessment based on the windowed bundle adjustment provides reliable results that are less conservative as we consider the uncertainty of the observations for each observation individually.

Furthermore, we evaluated the window length. While large window lengths are preferable for the windowed bundle adjustment and therefore need stably tracked features, the delay of the processed localization frame will increase. As we predict the real-time pose based on the localization frame pose, significant delays negatively influence the real-time pose estimate as the uncertainty increases. Hence, a balanced window length is vital.

In a direct comparison in Part 8.1.2.3, we illustrated that the MLE uncertainty assessment assuming Gaussian error distribution provides too optimistic results. We show that the $99.9\,\%$-confidence ellipsoid does not contain the correct pose for $99.9\,\%$ of the cases. The interval-based estimate, in contrast, is $100\,\%$ reliable as it always contains the true pose in our experiments. However, it is significantly more pessimistic as it accounts for higher drift.

In the runtime evaluation in Subsection 8.1.3, we showed that the visual odometry module is generally real-time capable. Nonetheless, due to the application of the windowed bundle adjustment in the back-end, the interval-based odometry computation is delayed, as discussed above. That is why we apply real-time pose prediction.

The main disadvantage of the visual odometry module is the delay due to the back-end. To avoid that, the visual odometry can be exchanged by odometric sensors such as wheel odometry or IMU. In future work, we can robustify the feature extraction by exploiting semantic information provided by methods such as [148–150]. For instance, we can avoid detecting features on objects that are likely to be dynamic (e.g. other vehicles).

Coarse Localization

The main goal of the coarse localization is to localize the vehicle under large uncertainties. The method consists of two parts: interval-based localization and bounded MCL. The interval-based localization provides the feasible set that satisfies two simple but always valid constraints. The first constraint is that the vehicle cannot be placed inside a building. The second constraint is that none of the filtered LiDAR points can lie beyond the facades. Based on the constraints, we introduced two contractors that gradually narrow down the feasible set. However, as we show in our evaluation in Section 8.2, the set can be very conservative with uncertainties of more than $10\,\mathrm{m}$, which is inappropriately large for the safe localization of the vehicle. Nonetheless, the feasible set makes it possible to reduce the location uncertainty of the GNSS of $40\,\mathrm{m}$ to a smaller region in our experiments. To improve the localization estimate, we augment the feasible set by a bounded MCL, spreading particles inside the feasible set to determine the most likely solution. Hence, the MCL approach is conditioned by the feasible set.

In our experimental results, we saw that the subpaving representation of the feasible set makes tracking the vehicle with multiple disconnected solutions possible, qualifying the approach for global localization tasks [18]. Nonetheless, to ensure the real-time capability of the method, we illustrated in the runtime analysis that an initial contraction of the feasible set is vital to reduce the subpaving as quickly as possible. That is why GNSS measurements, although they can have high uncertainties in urban environments, are essential – especially at the beginning. Since the vehicle moves while GNSS measurements continuously contract the feasible set, the rotation uncertainty can be reduced quickly. Only a small subset of feasible rotation angles remain consistent with the vehicle movement. We also showed that the approach can only work properly in urban environments with a high building density. In the case of a sparse map, neither the No-Overlap Contractor nor the No-Cross Contractor will reduce the feasible set.

Although the feasible set is very pessimistic, we demonstrated in Subsection 8.2.1 that the ground truth pose is always inside the feasible set. Consequently, the feasible set is reliable. Compared to the AMCL, our bounded MCL converges faster and has lower tracking errors while augmenting the feasible set with the most likely result. The coarse localization only

needs a few particles due to aggressive resampling that enables fast convergence. However, this comes with the cost that the localization track jumps and error spikes can occur, as pointed out in Subsection 8.2.2. Due to the deterministic property of the interval method, the hybrid approach shows better repeatability of the results. While particle depletion is a big problem for classical MCL approaches, our hybrid method can address the issue as the feasible set tracks the region where new particles can be spread.

A disadvantage of the coarse localization is that many parameters must be tuned. In particular, the LiDAR filter can admit points that violate the No-Cross assumption if not well tuned. If the No-Cross constraint is violated, the feasible set loses the guarantees to contain the correct pose.

In the current implementation of the bounded MCL, we only consider the simple beam-end model. In the future, more sophisticated observation models in the MCL can be used that employ machine learning approaches as presented in [93] and [148]. Furthermore, our method is only evaluated and tested in this work based on building maps. However, the coarse localization can also use other maps, such as street maps, to restrict the feasible region where the vehicle has to be located. In combination with other maps, other constraints can be included in the CSP by including more contractors. This will further reduce the pessimism of the feasible set and increase the convergence of the bounded MCL as it has to cover a smaller feasible set.

Refined Localization

The refined localization refines the coarse localization estimates by matching locally seen facades to the facades on the map. Therefore, the highest weighted particle of the coarse localization is used to perform the matching. The refined localization consists of two parts: a set-membership-based localization and a bounded optimization. While the set-membership-based localization determines the set consistent with the local matching to the facades, the bounded optimization provides the most likely pose that satisfies the matching in the least squares sense. The optimization is restricted to the consistent set during the optimization iterations. We presented the evaluation of the refined localization in Section 8.3.

Compared to the feasible set, the consistent set is significantly less pessimistic. The compact representation of the consistent set by a polygon and an orientation interval reduces the wrapping effect. However, the problem with the refined localization is its dependency on the bounded MCL results of the coarse localization. Hence, the initialization of the refined localization can differ in each run for the same dataset due to the random component in the bounded MCL.

Although the consistent set contained the correct pose in most frames, it does not always enclose the ground truth. Hence, the set-membership approach can be corrupted by incorrect associations. Consequently, the consistent set is less pessimistic but comes with the cost of being unreliable. It represents the counterpart to the feasible set that is very reliable but more pessimistic. That is why we introduce in the HyPaSCoRe Localization an alternative heuristic measure for reliability, to use the consistent set to reduce the pessimism of the feasible set in the cases where we can be sure that the consistent set is trustworthy. The consistent set provides the bounds for the bounded optimization and prevents the optimization from divergence. However, the bounds are only valid if the matching is correct. In the case of

incorrect matching, the consistent set may hinder the optimization from converging to the correct result. This happened in rare cases because we associate conservatively.

When we compared the author-collected dataset results with the KITTI results, we encountered a better performance for the facade association for the scanning LiDAR sensor with 360° FoV. We saw that the localization accuracy depends on the visibility of the facades and how well they are matched to the map facades. In the case of large occlusions of facades, the uncertainty rises, and the consistent set stays large.

The refined localization's main problem is that each individual facade's error bounds must be known. Unfortunately, this error is typically unknown, and assumptions must be introduced. While overestimating leads to overly pessimistic results, underestimating the error can lead to corrupted consistent sets that will not contain the correct poses.

The nearest neighbor approach can be augmented in future work by more sophisticated matching algorithms of local LiDAR data and map facades. For instance, the semantic segmentation of point clouds provided by machine learning approach such as [148] can improve the association. Alternatively, a set-membership-based association scheme can be implemented, that combines the consistent set contraction and association. Therefore, the constellation contractor presented in [144] should be considered in future works.

HyPaSCoRe Localization

We evaluated the whole localization system in Section 8.4. The main objective of the whole system is to combine the feasible set and the consistent set so that we maintain the integrity of the feasible set and reduce the pessimism by contracting it to the consistent set for those cases where the consistent set is staged to be reliable. Therefore, we introduced a way to assess the reliability heuristically. Furthermore, the windowed bundle adjustment makes direct localization only possible at a delayed time. However, to determine the vehicle pose in real-time, we evaluated the real-time pose prediction and compared the results to the localization time estimates.

Although the prediction in real-time has higher uncertainty as it accumulates the localization uncertainty of the processed frame and the frame-to-frame odometry, the absolute error is not significantly higher than at localization time. That also means that the uncertainty estimate is more pessimistic than it has to be. It is remarkable that for an accurate map such as the LOD2 and a good observation of the facades, we obtain an absolute error of less than $1\,\mathrm{m}$. We achieve this high accuracy only by considering the surrounding buildings showing enormous potential for localization of highly automated and autonomous vehicles. However, if facades are barely seen, the uncertainty becomes very large. Generally, this is the correct behavior of the HyPaSCoRe Localization since the absence of information means that the observed state of the vehicle becomes more uncertain, which signals that the absolute error of the vehicle location may increase. If the uncertainty is inappropriately high and makes safe operation impossible, we can detect that state and warn the passengers to intervene or stop the vehicle. The more differently oriented facades are visible and matched, the better the HyPaSCoRe Localization performs. However, not only the quantity but also the quality of the used maps significantly impact the pose estimation of the vehicle, as the comparison between the OSM and LOD2 showed.

In our evaluations, we showed that the feasible set always contains the correct pose (at least for the author-collected dataset with reliable ground truth), which underlines the high reliability and integrity of the estimate. On the one hand, it comes with very pessimistic uncertainties. On the other hand, the refined localization provides less pessimistic but also less reliable pose estimates. We demonstrated that we could reduce the large pessimism of the feasible set using the consistent set for those frames at which the consistent set is staged as reliable. We validated the integrity of the feasible set with the Standford-ESA integrity diagram. The contraction of the feasible set reduces the computation overhead as the subpaving becomes significantly smaller, and we gain computation time. Furthermore, we avoid the unnecessary spreading of particles as we only concentrate on the smaller consistent set. However, the critical part here is the reliability assessment of the consistent set. As presented in Subsection 8.4.2, we contract the feasible set to the consistent set if the refined localization track is old enough and if differently oriented facades are visible. This metric to measure reliability is just a heuristic that works well in practice for the datasets we considered in our evaluation. However, this metric does not guarantee that the consistent set is reliable regarding the correct pose. Hence, other reliability metrics need to be considered and evaluated in future work. For instance, the reliability can be estimated by introducing cross-validation of the pose estimate in different maps with other landmarks like road markings.

We showed that other, more appropriate results could be tracked by allowing switches and resets of the refined localization track. In the case of drift or a wrong track, we can always switch to the correct pose, which we detect with the bounded MCL in the feasible set in the coarse localization. On the downside, this comes with the cost that switches, and resets lead to jumps in the localization estimate.

The HyPaSCoRe Localization only provides accurate pose estimations after the initialization of the refined localization. Until then, the HyPaSCoRe Localization only provides the highly uncertain but reliable feasible set and the particles of the coarse localization. As the experimental results revealed, the initialization highly depends on the map structure and the performance of the bounded MCL. The disadvantage of the HyPaSCoRe Localization is that the vehicle has to generally ride a significant distance until the refined localization is initialized. Furthermore, the vehicle must move so that the feasible and consistent sets are continuously contracted.

Another weakness of the HyPaSCoRe Localization is the large number of empirically tuned parameters. Although we chose the same parameters for all datasets, they may perform poorly in other scenarios. Expert knowledge is necessary to fine-tune the parameters. Furthermore, a good knowledge of the sensor and map uncertainties is necessary to bound the error appropriately. While an overestimation provides unnecessarily pessimistic results, underestimating the error may corrupt the uncertainty computation of the feasible and consistent set. Finally, we showed that our hybrid localization approach is real-time capable since each component is carefully engineered and utilizes an appropriate multi-threading.

In future work, other maps that contain, for example, a terrain model, road signs, and guide posts can be integrated into the HyPaSCoRe Localization to improve the robustness and accuracy of the localization. To avoid the localization delay, the visual odometry can be exchanged by other dedicated sensors such as wheel odometers and IMU.

10 Conclusions

This thesis aims to exploit the cooperation between probabilistic and set-membership-based localization approaches. Whereas the significant part of State of the Art uses probabilistic or set-membership approaches, a new hybrid method called HyPaSCoRe Localization is introduced that combines both approaches into one unified method. The HyPaSCoRe Localization determines a robot's pose in a building map in real-time and satisfies high integrity standards while being less pessimistic on the uncertainty estimation.

The HyPaSCoRe Localization focuses on urban canyons. The main problem of urban regions is that GNSS data can become highly inaccurate as multi-path effects corrupt the GNSS-based location estimates. Our approach can cope with those large uncertainties. It can provide a feasible set of poses that encloses the correct pose and provides a maximum likely pose that best fits the local LiDAR measurements to the building map.

As shown in Figure 10.1, the HyPaSCoRe Localization consists of three main modules: visual odometry, coarse localization, and refined localization. While the visual odometry provides the relative motion of the vehicle, the coarse localization uses the building map and the relative motion of the vehicle to narrow down the feasible set of vehicle poses. Therefore, globally valid constraints on the vehicle pose and the observations are exploited. The refined localization refines the coarse localization results by introducing the consistent set as a subset of the feasible set that considers the most likely point-to-facades association of the local LiDAR points to the surrounding buildings. If the consistent set is staged as reliable, the pessimistic feasible set is contracted to the consistent set. Consequently, the HyPaSCoRe Localization provides the set of feasible poses representing the localization uncertainty and the most likely pose estimate determined by the refined localization.

In the introduction of this thesis, we raised three research questions. In conclusion, let us return to them and examine more closely what answers we have found in this work. The central research question of this thesis was:

Is there a symbiotic relationship between set-membership and probabilistic approaches that can be exploited to improve the robot localization estimation and the uncertainty assessment?

In the light of this work, we can answer this question with yes. In general, set-membership approaches aim to provide sets enclosing the solution. Consequently, they are good at restricting the solution set. In contrast, probabilistic approaches asses the solution by employing probability distributions. However, the major part of the State of the Art assumes Gaussian error distribution. The estimation simplifies to computing the mean solution and the covariance

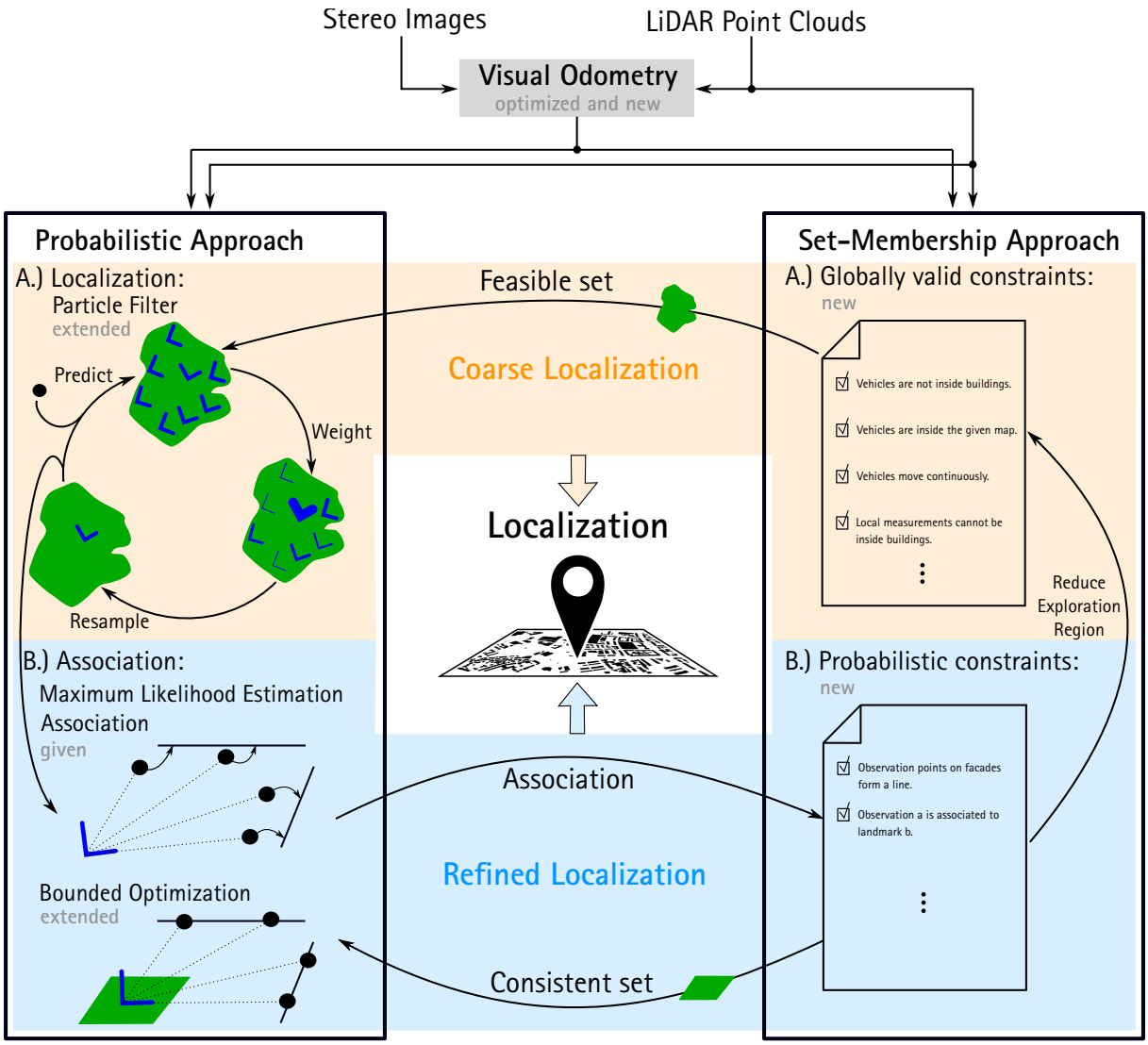


Figure 10.1: Method overview to the HyPaSCoRe Localization.

estimate. While the mean is often a good guess and is close to the correct pose, the covariance estimate that conveys the uncertainty information of the mean solution often underestimates the true error, which makes safe operation almost impossible, as the probabilistic method provides misleading information on the error of the estimate (cf. Standford-ESA integrity diagram in Figure 1.1). In our experiments in Part 8.1.2.3, we found that the $99.9\,\%$-confidence regions determined by a least squares probabilistic approach does not enclose the correct solution for $99.9\,\%$ of the cases which highlight the underestimation of the true error. Furthermore, typical probabilistic approaches, such as the particle filter or the MLE, tend to diverge in the case of incorrect data associations. In those cases, set-membership approaches are beneficial: As they restrict the solution space, the probabilistic approach can only operate within the defined set.

This observation leads us to the second research question:

How can set-membership approaches be used to robustify probabilistic approaches?

By restricting the solution set to the feasible region indicated by the set-membership approach, the probabilistic approach is restricted and is prevented from divergence. It becomes more robust since outliers that lead to solutions outside the feasible set can be identified and excluded.

This philosophy of hybridization is used in the coarse localization and the refined localization according to Figure 10.1. In the coarse localization, the interval-based localization restricts the bounded MCL to the feasible region. The bounded MCL converges up to four times faster compared to the State of the Art AMCL according to Table 8.2.2 as it concentrates on a subpart of the whole map, as illustrated in the top left part in Figure 10.1. The refined localization uses the consistent set to restrict the bounded optimization and therefore bounds the maximal possible error as evaluated in Subsection 8.3.2. As visualized in Figure 10.1, this type of hybridization makes the set-membership part the crucial component, as it restricts and controls the probabilistic part. Accordingly, the crucial part of the coarse localization is the interval-based localization on the top right side of Figure 10.1.

However, it is different for the visual odometry and the refined localization. Up to now, we have only discussed hybridization in the sense that the set-membership approach controls the probabilistic approach. But what about the other way? This leads us to the third research question:

How can probabilistic approaches be used to reduce the pessimism of set-membership approaches?

Probabilistic approaches make maximum likely associations possible. Although those associations may occasionally be unreliable, they often provide valuable information. If we can verify the reliability of the association, this information can be used to draw further constraints that can be used to further contract the solution set. Accordingly, by introducing further constraints from probabilistic approaches, we can reduce the pessimism of set-membership approaches. This philosophy of hybridization is predominantly used in the visual odometry and the refined localization modules. In the visual odometry, the windowed bundle adjustment preselects those observations used in the interval-based odometry estimation. In the refined localization, the pose particle with the highest weight of the coarse localization is used to associate the local LiDAR measurements to the facades as illustrated in the bottom left in Figure 10.1. Based on those associations, constraints are formulated to determine the consistent set, as illustrated in the bottom right in Figure 10.1. If the consistent set is reliable, we contract the feasible set of the coarse localization that significantly reduces the pessimism of the set-membership localization estimation. This hybridization approach makes the probabilistic part the crucial component as it controls the constraints used in the set-membership approach. The crucial part of the refined localization is the association illustrated in the bottom left part of Figure 10.1.

The experimental results showed that the HyPaSCoRe Localization maintains integrity with the feasible set while providing accurate, most likely point-valued solutions. While we validated that the true solution is always contained in the feasible set, the average positioning error is between $0.8\,\mathrm{m}$ and $1.8\,\mathrm{m}$ for the author-collected datasets evaluated with reliable ground truth. It is remarkable that when utilizing a precise map like the LOD2 and if a good observation of the facades is possible, we achieve an absolute error of less than $1\,\mathrm{m}$. This impressive level of accuracy is solely accomplished by taking into account the surrounding buildings, which demonstrate immense potential for localizing highly automated and autonomous vehicles. Nonetheless, the critical discussion of the results revealed some weaknesses that need to be improved in future works. Although the experimental results showed that the uncertainty assessment in the visual odometry provides appropriate and less conservative results, the method highly depends on the correct convergence of the windowed bundle adjustment, which

we cannot guarantee. Furthermore, the windowed bundle adjustment introduces a delay of the localization frame, so a pessimistic prediction of the pose in real-time is necessary. The visual odometry module can be exchanged or augmented by additional sensors, such as wheel odometers, to avoid those disadvantages.

The coarse localization provides the reliable feasible set and particles representing the most likely poses within the feasible set. In the scope of this work, we only consider a simple beam-end model on the bounded MCL. In the future, more sophisticated observation models can accelerate convergence and lead to a faster initialization of the refined localization.

The refined localization provides the most likely pose within the consistent set. The point-to-facades association uses a simple nearest-neighbor approach. In future works, semantic segmentation of LiDAR points can improve and robustify the association. Furthermore, the assessment of the reliability of the consistent set is heuristically determined by the age of a track. By introducing other independent maps, cross-validation can be applied to assess the reliability of a pose estimate. Although the pessimism of the error estimate is reduced in the HyPaSCoRe Localization , the integrity assessment showed a significant overestimation of the error. We identified that the high pessimism comes from parts of the building maps where barely any facades are seen or only very similarly oriented facades are observed. The localization can be improved and the pessimism reduced by using different maps such as road, lane-marking, and sign maps.

In summary, this thesis shows that exploiting the symbiotic relationship between probabilistic and set-membership approaches reduced the pessimism of the set-membership approaches and robustifies probabilistic approaches while maintaining integrity. However, this work only considers building maps. Extending the approach to different maps and using multiple maps to cross-validate the localization result will increase the robustness and reduce the pessimism of the HyPaSCoRe Localization.

Appendices

A Basics to Probabilistic Approaches

In this chapter, we briefly summarize the concept of Bayes filters and introduce the Gaussian implementation of the Bayes Filters known as the Extended Kalman Filter (EKF). The non-parametric implementation of the Bayes filter is the particle filter introduced in Subsection 2.1.2.

A.1 Bayes Filters

The Bayes Filter is the most general algorithm that calculates the probability distribution of the current state of the robot based on the previous state, the measurements, and control data. The Bayes Filter recursively computes the belief $\text{bel}(x_t)$ at time t from the belief $\text{bel}(x_{t-1})$ at time $t-1$. A belief is defined in probabilistic robotics as posterior probabilities over state variables conditioned on the available data. As a consequence, the belief

$$\text{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \tag{A.1}$$

is just an abbreviation of the posterior probability on the state of the robot that we are interested in. The Bayes Filter algorithm is depicted in Algorithm 10.

Algorithm 10: Bayes Filter

Data: $\text{bel}(x_{t-1})$, u_t, z_t
Result: $\text{bel}(x_t)$
1 **for** *all* x_t **do**
2 $\quad \overline{\text{bel}}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \text{bel}(x_{t-1}) dx_{t-1}$;
3 $\quad \text{bel}(x_t) = \eta p(z_t \mid x_t) \overline{\text{bel}}(x_t)$;
4 **end**

The Bayes Filter contains two essential steps. The first step presented in line 2 is the so-called prediction step and predicts the belief on the current state using the previous state and the control input. Note that this step does not involve any sensor measurements and therefore represents an a priori probability estimate. The second step presented in line 3 is the measurement update. The Bayes Filter multiplies the predicted belief $\overline{\text{bel}}(x_t)$ by the probability that the measurement z_t may have been observed for all hypothetical posterior states x_t. Since the product does not generally result in probabilities, the values may not integrate to 1. That

is why a normalization factor η is introduced that maps the product to probabilities that we interpret as the final belief bel(x_t).

The Bayes Filter is a very general algorithm to calculate beliefs. That means this filter approach applies to any arbitrary PDF. However, the integral in line 2 and the product of probabilities in Algorithm 10 may become very heavy in computation. If we assume a normal distribution for all involved quantities, the Bayes Filter simplifies to the well-known Kalman Filter. We will present the more general Extended Kalman Filter (EKF) as an essential representative of the Gaussian implementation of the Bayes Filter (cf. Section A.2). The Bayes Filter can also be implemented with non-parametric distributions. Instead of using an analytical description of the PDF, the probability distribution can be represented by particles. Implementing the Bayes Filter with a particle representation of the probability distribution is the so-called particle filter introduced in Section 2.1.2.

A.2 Extended Kalman Filter

The Kalman Filter is the Gaussian version of the Bayes Filter. The classical Kalman Filter is an optimal state estimator if the observation functions are linear, the next state is a linear function of the previous state, and all random variables follow a normal distribution. If that is the case, the Kalman Filter is a very efficient filter approach since the parameters of the resulting Gaussian can be computed in closed form. However, in practice, neither the state transition nor the observation functions are linear, which poses a significant limitation to the classical Kalman Filter. Therefore, the Extended Kalman Filter (EKF) relaxes the linearity assumption and makes applying to non-linear problems possible. Since the EKF is a strict generalization of the classical Kalman Filter, we will only introduce the EKF here.

The assumption is that the state transition and the measurement are described by non-linear functions

$$x_t = g(u_t, x_{t-1}) + \epsilon_t, \tag{A.2}$$

$$z_t = h(x_t) + \delta_t, \tag{A.3}$$

respectively. The state transition is corrupted by a Gaussian random vector ϵ_t that introduces the uncertainty. Similarly, δ_t describes the measurement noise. The main problem in the EKF is the non-linearity of the involved functions: Although the measurement uncertainty and the initial state uncertainty can be Gaussian, the belief can no longer be Gaussian. Performing the belief update exactly is usually impossible for non-linear functions g and h – there are no closed form solutions. Figure A.1 illustrates this problem. If we propagate a normally distributed random variable through a non-linear function, the output PDF cannot be a normal distribution. As shown in Figure A.1, only if the function is linear a normal distribution is propagated to a normal distribution that can be computed in closed form. That is why the EKF performs a local linearization: Since a linear function maps a normal distribution to a normal distribution, the EKF approximates the non-linear function at the mean point of the input. Hence, using linearization, the output PDF can be determined in closed form.

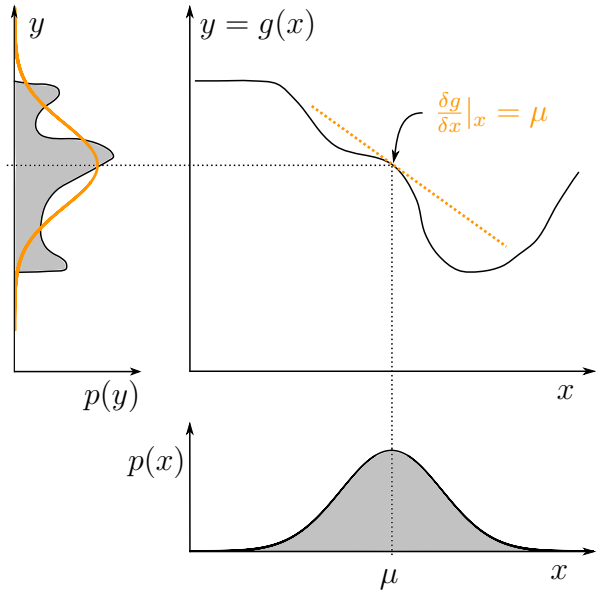


Figure A.1: Illustration of the linearization in the EKF algorithm based on [18]. The probability distribution of x is propagated through a non-linear function $y = g(x)$. The input variable is normally distributed. The corresponding PDF is drawn at the bottom. The exact PDF passed through $g(x)$ to the output y is not described by a Gaussian and is illustrated on the left. To generate the Gaussian approximation of the uncertainty, the EKF linearizes $g(x)$ at the mean of the original Gaussian. An orange dashed line illustrates the linear approximation. By propagating the original Gaussian through the linear approximation, the output PDF is also a Gaussian and is visualized by a solid line.

The EKF algorithm is depicted in Algorithm 11. Note that the EKF maintains the classical structure of a Bayes Filter that consists of a prediction and correction step. In the prediction step in lines 1 to 3, the predicted state only considers the previous state and the control command. In line 2, the state transition function is linearized. A Taylor approximation with the first and second terms is introduced in this case. Consequently, G_t is also called the Jacobian of the transition function concerning the state. In line 3, the covariance of the previous state Σ_{t-1} is propagated via the Jacobian G_t to the prediction covariance $\overline{\Sigma}_t$. R_t is the covariance of the random variable ϵ_t. The correction step incorporates the sensors measurements z_t into the state estimation. However, the Kalman gain (cf. Algorithm 11 line 5) determines whether we should rely more on the prediction or the sensor measurement. Note that the covariance of the measurement noise is Q_t.

The EKF provides a closed-form solution for the state estimation problem for dedicated linearization points. This makes the EKF very efficient in computation. The EKF assumes all involved uncertainties to be Gaussian distributed. In practice, the EKF performs well if the error is small and the linearization is a good approximation. However, highly non-linear functions may lead the EKF to diverge as the linear approximation is insufficient. In this case, the EKF may significantly underestimate the state uncertainty defined by the covariance matrix Σ_t as the linearization error is not considered in the state estimation. Furthermore, the EKF cannot handle non-Gaussian PDFs and therefore is very limited. Although there are extensions like, for instance, the Multi-Hypothesis EKF that can deal with mixtures of Gaussians, in the

Algorithm 11: Extended Kalman Filter

Data: μ_{t-1}, Σ_{t-1}, u_t, z_t
Result: μ_t, Σ_t
// Prediction step
$\overline{\mu}_t = g(u_t, \mu_{t-1})$;
$G_t = \frac{\delta g(u_t, x_{t-1})}{\delta x_{t-1}}|_{x_{t-1}=\mu_{t-1}}$;
$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$;
$H_t = \frac{\delta h(x_t)}{\delta x_t}|_{x_t=\overline{\mu}_t}$;
// Kalman gain
$K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$;
// Correction step
$\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$;
$\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$;

literature, the particle filter has proven to be a good alternative in the case of arbitrary PDFs and strong non-linearities but coming with the cost of higher computational effort.

B State of the Art – Visual Odometry and SLAM

The main goal of visual odometry is to estimate the relative motion of a robot equipped with one or multiple cameras concerning a defined reference frame. This frame is typically selected at the beginning of a trajectory we seek to determine. Nister coined the term visual odometry [151], who compared the image sensor-based incremental motion estimation to wheel odometry. As a result, wheel and visual odometry pursue the same goal, however, with different sensors.

The problem of recovering the relative camera poses, and the 3D structure from a set of images is known in computer vision as Structure-from-Motion (SfM) since the early 80ties starting with pioneering works such as [152]. Visual odometry is a particular case of SfM. While SfM typically refines camera poses and the structure in offline optimization (i.e. bundle adjustment), visual odometry focuses only on estimating the camera poses sequentially and in real-time [153].

Visual odometry can be divided into monocular and multi-camera approaches depending on how many cameras – that have a substantial overlap of the FoVs – are used. In the scope of this work, we are using a stereo camera setup that uses two cameras. The main advantage of using multi-view visual odometry compared to monocular approaches is that the motion scale is fully determined. In the case of monocular visual odometry, one camera only provides bearing information on the scene. As a consequence, the relative motion of the vehicle in the monocular case can only be determined up to a scale factor. To overcome this problem, other sensors, such as IMU or range sensors, are typically employed to determine the scale. Nonetheless, algorithms that are used for monocular visual odometry can be extended to the multi-view case. This section will consider monocular and stereo-vision approaches for visual odometry [153].

The main problem of dead reckoning approaches is the accumulation of errors when the motion is determined incrementally frame by frame. A classical approach to counteract this problem is windowed bundle adjustment. SfM is not online capable. However, it typically has high accuracy due to a bundle adjustment. Typically, a sliding window bundle adjustment is employed to increase the accuracy of visual odometry. While it decreases the drift, the sliding window bundle adjustment also has a higher computational cost. The drift can also be reduced by combining it with other sensors [153].

Although this section focuses on visual odometry, it is worth mentioning that there is a parallel, closely related research field: Visual Simultaneous Localization and Mapping (V-SLAM). The goal of SLAM, in general, is to obtain a globally consistent robot path. This goal also implies that the approach needs to keep track of a map of the environment, although in this

case, the map is not needed per se [153]. In SLAM, the map is used to detect when the robot revisits already seen places and to draw additional constraints to further reduce the drift in both the map and trajectory. Integrating place recognition techniques into estimating the map and the trajectory is called loop closure and poses one of the key components of SLAM.

In contrast, visual odometry aims at recovering the trajectory incrementally – potentially optimizing only over the last n poses of the trajectory (sliding window bundle adjustment). However, we can interpret the sliding window optimization as equivalent to building a locally consistent map in SLAM. However, the philosophy differs: While visual odometry only cares about the local consistency of the trajectory, SLAM is concerned with the global map consistency [153]. Nonetheless, visual odometry is typically used as one building block of complete SLAM algorithms to obtain the incremental motion – while further building blocks like loop closing and global optimization extend visual odometry to V-SLAM. Consequently, to have an overview of the existing literature on visual odometry, we will also introduce V-SLAM algorithms and visual dead reckoning approaches, mainly focusing on the visual odometry component.

In visual odometry, there are probabilistic, interval-based, and hybrid approaches. In the following, we will introduce relevant State of the Art visual localization approaches pointing out the approaches' differences, advantages, and disadvantages.

B.1 Probabilistic Approaches

Similar to the already introduced probabilistic robot localization algorithms in Section 3.1, the major part of visual odometry and V-SLAM research model uncertainties with probabilistic means. Due to the convenient properties of the normal distribution, the Gaussian assumption has become very classical in this field, which makes the EKF and MLE methods applicable to visual odometry. However, due to the sequential nature of visual odometry, other Bayesian filtering methods – such as a modified version of the particle filter – can be applied to this problem. With rising computation power, MLE methods have proven their accuracy in visual odometry and pose the current State of the Art. One of the main assumptions that make MLE approaches so efficient is the Gaussian assumption. Only this assumption on the measurements makes it possible to solve the MLE problem by a least squares formulation usually solved by a numerical optimization method such as the LM algorithm [10, 23]. The following introduces some relevant implementations of the different estimation strategies.

B.1.1 Extended Kalman Filter

The core assumption for applying the EKF to visual odometry is that all our measurements are zero-mean Gaussian distributed and that two arbitrary measurements of the same state are always statistically independent. Note that this assumption is not necessarily satisfied by camera observations. External factors like lighting conditions or dynamic objects may corrupt measurements of the same landmark in the scene so that the measurements are not statistically independent anymore as the same perturbations affect them. Nonetheless, the EKF paved the way for one of the first real-time capable visual odometry implementations.

Davison et al. present in [154] a real-time capable monocular SLAM method applying full covariance EKF for probabilistic state estimation. The camera poses, velocity and visual landmarks define the state. The natural visual landmarks are detected initially by a Shi and Tomasi operator [155]. Instead of only storing the detected points, the authors store image patches that contain the salient features and the orientation of the approximated surface. To reobserve the detected feature in subsequent images, the patch can be projected from 3D to the image plane to produce a template for matching. A warped version of the original patch taking the patch orientation with respect to the camera pose is used for matching by normalized cross-correlation. However, since the approach only uses a monocular camera, the 3D position of a feature cannot be determined as the scale is not fixed. The authors solve this problem by initially providing features with known positions and appearance. This makes fixing the scale during initialization possible.
For the prediction step, the authors assume a constant velocity model. The update step is based on the patch correlation-based matching of the mapped features to the actual image regions. While the uncertainty is estimated by a full covariance matrix taking all landmarks into account, the EKF-based approach can track the correlation between features. However, this comes with the cost of more states that need to be estimated in the EKF. That is why the proposed method is not arbitrarily scalable to larger scenes since more landmarks directly lead to more computation, and the method becomes not real-time capable anymore. Further, the authors do not update an initialized feature patch to prevent drift. However, this makes the method less resistant to lighting changes in the scene, and the scale can drift over time if not already mapped features are not reobserved for a longer period.

Paz et al. tackle the problem of an increasing state vector by dividing the scene into multiple sub-maps [156]. Instead of performing EKF-SLAM for the whole environment and keeping track of all correlations, the authors approximate the state estimation by producing multiple independent local maps. This enables to bound the maximum number of states for the EKF-SLAM and therefore does not suffer from quadratically rising computation costs. However, the correlation between local maps is not considered by dividing the environment into limited sub-maps, so valuable information is unused in the state estimation. Consequently, the proposed approximation requires the assumption that the sub-maps are statistically independent – which cannot be the case as the scene is explored sequentially. To partially tackle this problem, the authors propose to define shared states between the EKF-SLAM sessions by inserting a limited number of further states representing common landmarks in both sub-maps. Hence, instead of requiring statistical independence, they can lose this restriction to conditional independence. The overhead slightly increases the computation. However, the correlation between the maps is at least partially modeled.
The authors use a stereo camera that overcomes the scale ambiguity of the monocular case. Nonetheless, the update step does not only take stereo features into account but also includes monocular observations from both cameras. Due to the correlation via the covariance matrix of the states, the monocular observations are fixed in scale by the stereo observations.
The main problem with this approach is that dividing the scene into smaller maps may amplify the drift. Consequently, sub-maps that may be created at different points in time may refer to the same region in the actual scene. Hence, further loop closure techniques are necessary, as

also proposed by the authors. However, correcting the trajectory and the map based on the loop closure constraints again results in a costly and time-consuming computation.

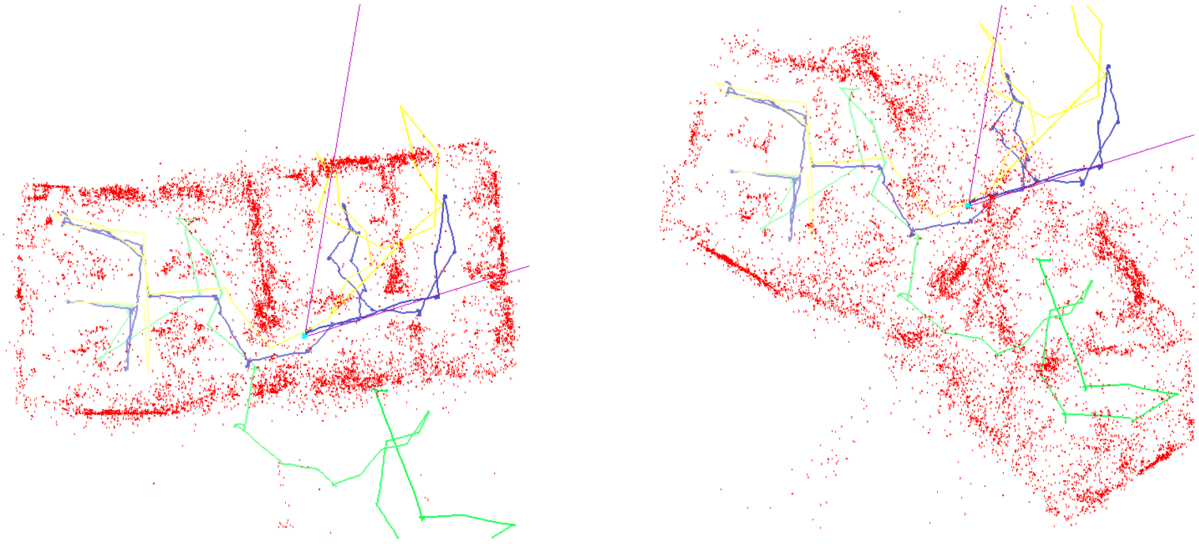
B.1.2 Rao-Blackwellized Particle Filter

Although the PF has proved to be a powerful Bayesian localization method that can cope with non-linearities as this filter is non-parametric, the straightforward implementation of the PF to visual odometry and V-SLAM is doomed to fail: Due to the large number of variables involved in describing the map and the robot's trajectory, the PF becomes inefficient [18]. However, there is a convenient property of SLAM problems that can be exploited to apply particle filters: The full SLAM problem, in which we assume to know the correspondences of landmarks and their observations at different points in time, possesses a conditional independence between any two disjoint sets of landmarks in the map, given the pose [18]. Consequently, if we know the robot's pose, we can deduce the location of each landmark independently of each other. Dependencies are only introduced through the robot pose uncertainty [18]. This observation paved the way to a factored version of the PF - the Rao-Blackwellized Particle Filter (RBPF) - that makes applying the PF to the SLAM problem feasible.

The core idea of the RBPF is to use the particles to represent the posterior over a subset of variables. The remaining variables are modeled with easy-to-handle parametric PDF-like Gaussian distribution by exploiting conditional independence. FastSLAM [157, 158] is one of the first and most well-known implementations of RBPF to SLAM with 2D LiDARs. FastSLAM represents the robot's pose by a set of particles. However, each of those particles tracks a map individually. Now, conditional independence comes into play: Since the individual map errors are conditionally independent for each of the particles, the mapping problem can be factored into many separate problems – one for each landmark per particle [18]. FastSLAM uses a low-dimensional EKF for each landmark estimation problem. Consequently, FastSLAM uses particles to represent the robot's pose while each particle carries its own map that is updated individually for each landmark using an EKF.

Sim et al. [159] apply RBPF to V-SLAM for indoor mobile robots equipped with stereo vision. The landmark estimates are derived from stereo images using Scale Invariant Feature Transform (SIFT) [160] features, and the motion estimates are based on visual odometry. SIFT features are detected in the left and right images and matched using epipolar geometry (rectified stereo image pairs) and SIFT descriptor similarity. The landmarks are distinguishable based on the SIFT descriptor. The authors determine the relative motion between two consecutive frames based on the tracked features to perform the prediction step for the particles. By minimizing the re-projection error of the 3D coordinates of the landmarks by applying the LM optimization algorithm [23], the authors compute the relative transformation that maximizes the log-likelihood of the relative pose. Figure B.1 shows the difference between maps that result from the RBPF-based pose estimation and the pure odometry-based map.

The observations model to weight the particles relies on the descriptor-based matching of the detected features and the already mapped landmarks. As each particle represents the vehicle's pose, the locally observed features can be transformed into the map frame. The authors define a weighting strategy for the particles that relies on the distance of the transformed feature and the mapped landmark. For each landmark, an EKF is utilized to correct and update its position



(a) Constructed map for the best sample at the end of the trajectory. The map is consistent, and the room structure is preserved. [159]

(b) Constructed map solely only relying on visual odometry. The map is corrupted by drift, and the room structure appears bent. [159]

Figure B.1: Constructed maps with different trajectories. The blue trajectory indicates the trajectory of the best sample, he green trajectory indicates the visual odometry measurements, and the yellow trajectory the robot's wheel odometry. [159]

in the map frame. The authors show that their approach consistently estimates the map and the trajectory, although the visual dead-reckoning is corrupted by drift as shown in Figure B.1 Wu et al. [161] improve [159] by introducing more mature landmark management using rebalancing trees. Furthermore, a robust initialization of the LM optimization algorithm with a RANSAC approach is introduced. A set of features are drawn from the seen features to compute the initial relative motion estimate. This approach improves the experimental results. However, the suggested method is not online capable due to the expensive SIFT feature detection and many particles necessary for convergence. Furthermore, it is not easily scalable to large outdoor scenarios.

B.1.3 Maximum Likelihood Estimation

MLE approaches were already introduced in the context of localization in Subsection 3.1.4. We saw that the MLE approach yields outstanding results if the initial solution is good enough so that the optimization can converge to the correct pose. Furthermore, the MLE approach also assumes that there only exists one unique and correct solution. That means the MLE approach must decide which solution it has to stick to in the case of ambiguities. Especially for the localization problem, the MLE approach can run into severe problems because the localization estimate may initially be very uncertain. Sticking to just one consistent solution may become problematic if this locally consistent pose is incorrect.

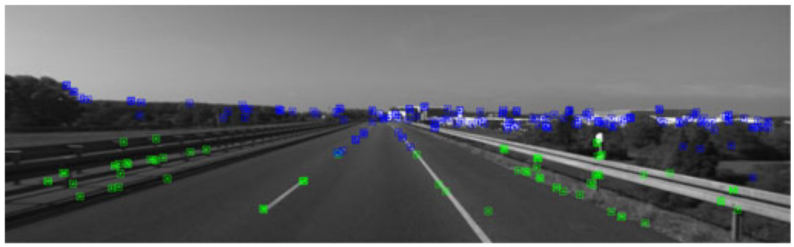
However, in this chapter, we are dealing with a topologically different problem: The goal of visual odometry is the computation of the relative motion of the vehicle. As a result, the incremental structure of the problem complies well with the strengths of an MLE approach:

The MLE approach provides for each time step the most likely relative motion that typically results in visual odometry into a comparatively low drift and therefore provides reasonable estimates.

For visual odometry computation, the Gaussian assumption in the context of the MLE approach has proven to be very convenient: Since many different but minor errors mainly influence the incremental pose estimation, the Central Limit Theorem justifies the Gaussian assumption. Consequently, the MLE problem under Gaussian uncertainty can be formulated as a least squares problem for which sophisticated solvers exist. In the case of non-linearities (i. e. rotation), incremental optimizers that perform local linearization often lead to acceptable results. However, as we will see, many methods do not consider the uncertainty estimate provided by the optimization approach since those estimates are often very optimistic and do not reliably consider the uncertainty.

In [139], Klein and Murray suggest dividing the tracking and mapping into separate working threads. While the tracking performs real-time operations providing visual odometry, the mapping part runs on a lower frequency. It performs global bundle adjustment based on keyframes, which form a subset of all frames to improve accuracy. This approach leaves enough computation resources available for the visual odometry to perform a windowed bundle adjustment online. Similar to [154], features are stored with a planar patch. However, to efficiently detect the points of interest, the authors apply the FAST corner detector [162]. By template matching, features observed in consecutive frames are associated. The relative pose is determined by numerical optimization of the re-projection error similar to [163]. Furthermore, the pose is refined by a windowed bundle adjustment. The authors show that this approach outperforms the EKF and RBPF for v-SLAM as it can cope with more landmarks and does not have to deal with multiple particles. Due to the parallel nature of the processing, the authors call their approach Parallel Tracking and Mapping (PTAM). Many follow up works [134, 164–166] have adopted this architecture. While the initial implementation was developed for monocular cases, follow-up works such as [165] extend PTAM to stereo vision avoiding the scale problem.

One of the most widely used v-SLAM implementations is ORB-SLAM, developed by Mur-Artal et al. [134] that builds upon PTAM. One of the main disadvantages of PTAM is that it can only operate in small-scale environments, lacks adequate loop closures for refinement, and has low invariance to viewpoint changes due to the template-matching approach. The authors improve PTAM by using ORB features [130] for tracking, mapping, relocalization, and loop closing (see Figure B.2a). The most interesting part of ORB-SLAM in the context of this work is the suggested tracking architecture that provides visual odometry information. A map with the seen features needs to be initialized to start the tracking. Since ORB-SLAM, as presented in [134], uses a monocular camera, an initialization procedure based on heuristics is suggested. However, ORB-SLAM was extended to RGBD- and stereo-cameras in [26] (ORB-SLAM2) so that the scale problems of monocular cameras can be avoided. If a map is initialized and the last frame is successfully tracked, the camera pose is predicted by assuming a constant motion model. As a result, a guided search of the map points detected in the last frame is performed. Based on this initial set of features, the predicted pose can be optimized by minimizing the reprojection error. The improved pose is then used to project the local map in the camera's FoV to search for more map point correspondences. The camera pose is finally optimized



(a) ORB feature detection. Points close to the camera are colored green, and far features are colored blue [26].

(b) Trajectory with many loops and the corresponding ORB feature map.

Figure B.2: ORB-SLAM detects ORB features in the camera image and determines the camera pose by tracking the features. The SLAM system performs loop-closures to improve the trajectory and map by global bundle adjustment [26, 134].

with all the map points found in the frame. As PTAM, if the number of tracked features drops significantly, a keyframe is inserted. When a keyframe is inserted, all the features newly detected in the frame are inserted as further map points. ORB-SLAM further improves the pose estimate and the map by loop closure constraints and global bundle adjustment (see Figure B.2b). The authors use the graph optimization approach g^{2}o presented in [20, 29].

Up to this point, we only considered a feature-based approach for visual odometry and SLAM. However, larger regions can be used to determine the camera movement instead of relying on distinct features in the scene. For instance, LSD-SLAM [166] is an appearance-based approach. This approach's core idea is to minimize the photometric error of aligned image regions instead of minimizing the reprojection error of distinct points in the scene as illustrated in Figure B.3. While the appearance-based approach is well suited for visual odometry, distinct features still need to be used for detecting loops. Another drawback of the photometric minimization approach is that a good initialization is necessary. Consequently, in the scope of our work, we will stick to the feature-based visual odometry approach due to the problems mentioned above of the appearance-based approaches and due to the high robustness of feature-based approaches as demonstrated in [26, 134].

Recent visual-odometry approaches improve the pose estimate by fusing data with further sensors. For example, Zhang and Singh present in [167] a visual-LiDAR odometry and mapping approach with low drift that operates robustly in real time called VLOAM. The online method starts with visual odometry to estimate the sensor system's ego motion and correct the motion-blurred point cloud of a continuously spinning LiDAR sensor. To accomplish that, the authors track three types of visual features with a monocular camera: depth-registered features, features with triangulated depth from multiple frames, and features without depth information. An equation system can be formulated based on the tracked set of features. An optimizer solves that in the least squares sense. Hence, Gaussian uncertainties are assumed. Using the ego-motion estimate from the monocular camera enhanced by the depth information from the previous LiDAR measurements, the current LiDAR scan be corrected by compensating the motion. The corrected point cloud is then added to the subsampled depth map, which

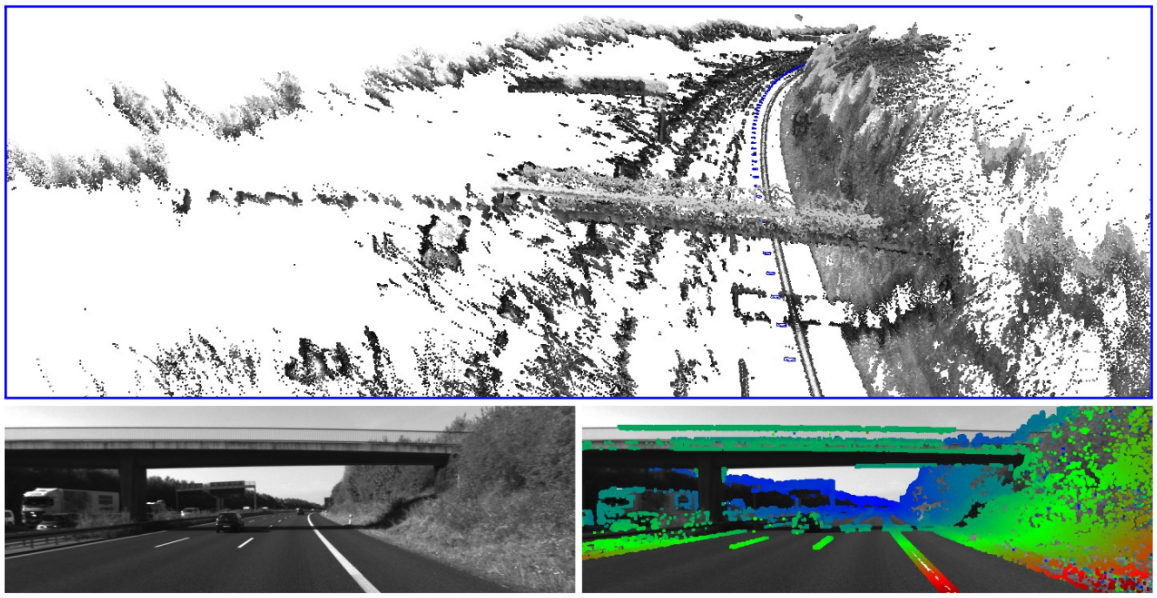


Figure B.3: LSD-SLAM. The visual odometry is computed minimizing the photometric error of regions in the image that are rich in texture [166].

associates depth information with the visual features. Shao et al. enhance VLOAM in [168] by a stereo vision system and an IMU for motion compensation for the LiDAR data.

Visual odometry and v-SLAM approaches using MLE achieve impressive results with low error compared to filter-based methods such as the EKF and the RBPF. However, most existing techniques focus mainly on determining the most likely map and leave open how to efficiently compute the uncertainties [169, 170]. Consequently, the MLE approaches only provide solutions without quantifying the estimate's trustworthiness. There are just very rare works that try to tackle this problem. The basic foundation of all MLE approaches that solve a least squares problem is the Gaussian assumption for all involved measurements.

Consequently, due to the construction of such approaches, the uncertainty of the estimated states – that means the robot poses and landmark locations – are modeled by multivariate normal distributions. To determine those marginal covariances, the system matrix of the least squares formulation as introduced in Subsection 2.1.3 needs to be inverted. However, this inversion operation is very intense in computation, as the least squares formulation may involve many poses and landmarks. This makes real-time computation impossible and therefore is often ignored. Nonetheless, rare works like [169] and [170] introduce approximations to accelerate uncertainty estimation. Alternative approaches, such as the Gaussian Belief Propagation, bypass this bottleneck, computing the estimates by sequential propagation of the state information to the connected states [171]. This approach exploits the property of SLAM problems that can be formulated in a graphical structure as illustrated in Figure 4.4. However, to use this approach efficiently, special Intelligence Processing Units (IPU) are necessary [172]. However, not only the computation time is a problem for the very popular MLE approaches regarding uncertainty estimation: If a state has many measurements, the covariance of the state will decrease as its estimate becomes more and more certain. Consequently, if there are two measurements for one state, while one measurement type generates more measurements than the other, the measurement type with more measurements will have a higher impact on the

state estimation. That means, in the long run, the measurement type with more measurements will determine the state and its uncertainty as it will overshadow the other, although it may be less accurate. To mitigate this problem, the input covariances are accordingly scaled, or the number of measurements must be modified accordingly. However, those workarounds can harm the solution and corrupt the uncertainty estimation.

To sum up, while MLE approaches provide, in practice, very impressive results, uncertainty estimation is often neglected. On the one hand, it poses a bottleneck for real-time applications. On the other hand, it provides overly optimistic results due to vanishing marginal covariance that relies on assumptions of zero-mean Gaussian error. Consequently, relying only on the Gaussian assumption is insufficient – alternative error models are necessary to cope with the abovementioned problems. Hence, we will consider interval-based visual odometry approaches in the following subsection.

B.2 Interval-based Approaches

Interval-based approaches represent the uncertainties for all involved variables by construction. In this section, we want to draw our attention to the basic principle that interval-based approaches follow: The goal of interval-based approaches is to cast the existing problem to a set of (possibly nonlinear) equations and inequalities. Those equations and inequalities are interpreted as constraints on the variables that encode the poses and landmark locations. Interval analysis provides the mathematical tools to propagate information between those variables using the constraints. That means, for instance, if we know the location and orientation of the very first pose – which is typically the case in SLAM – we can use the constraints to deduce the following poses taking the movement constraints based on the dynamics into account. Furthermore, the landmark locations seen from the first pose are determined via observation constraints. Since all variables are defined in the interval domain, we can also consider the uncertainty. As a consequence, interval-based approaches are very similar to graph-based MLE approaches. Here, the goal is to set up the constraints in one large equation system and solve that equation system. However, both approaches solve the set of equations seeking different philosophies: MLE approaches are only interested in the most likely configuration of the variables and typically solve the constraints in the least squares sense by an iterative approach (e. g. LM). However, interval analysis considers the set of all *feasible* solutions consistent with the constraints.
Passing information between connected variables via constraints is not unique to interval-based approaches. For instance, Gaussian Belief Propagation pursues the same approach – instead of propagating intervals, Gaussian beliefs (normal distributions) are passed as information between the connected variables [171–173] to infer the variable's state and uncertainty. Consequently, interpreting and storing the involved variables in a graphical structure as presented in Subsection 2.1.3 is a good way to formalize the SLAM problem as it reflects the nature of the problem.

Jaulin presents in [174] an interval-based SLAM approach with underwater sonar images enhanced by a loch-Doppler, a gyrocompass, an altimeter, and a barometer. The sonar images detect fixed seamarks on the seabed, similar to image features for classical cameras. A side scan sonar image is illustrated in Figure B.4. The goal in [174] is to map the seen seamarks

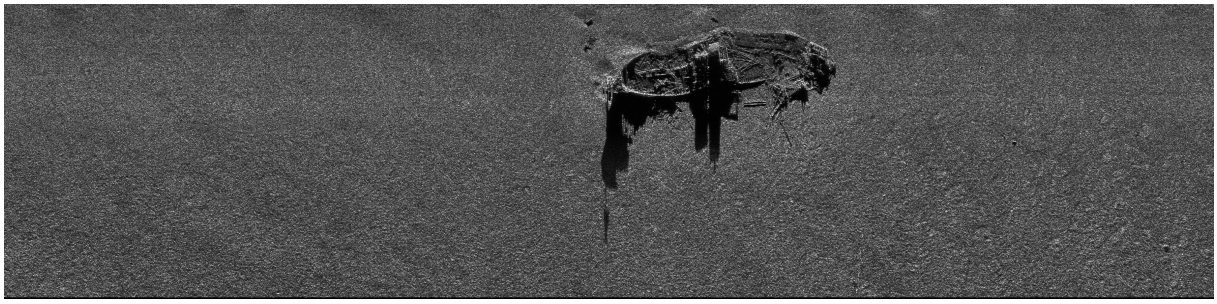


Figure B.4: Sonar image. Just one side of the side-scan sonar is shown. An old shipwreck on the seabed is detected. Image credit: Simon Rohou.

and to reconstruct the trajectory. Although the sensors on board well measure the submarine's motion, the determined relative motion still suffers from drift. As GPS signals do not propagate underwater, GPS-based correction is not possible. On the one hand, the authors use the seamarks to reduce the drift. On the other hand, using interval analysis, the authors can measure the submarine's drift by considering the sensor properties and providing the feasible set of locations of the seamarks on the seabed. To accomplish that, Jaulin considers the following constraints: An evolution equation that considers the submarine's velocity and orientation determines the submarine's movement. Furthermore, if the submarine is on the surface, it receives GPS signals. Consequently, two further observation constraints on the submarine's position in a global frame can be drawn for the mission's beginning and end. The last set of equations is based on the seamark observations in the sonar images. By performing interval propagation through all constraints that connect the pose and seamark variables, the authors can solve the SLAM problem as shown in Figure B.5. Note that the resulting seamark locations (black boxes) and the pose at different points in time are described by intervals inherently providing the estimates' uncertainties. However, this approach solves the SLAM problem in an offline fashion. Furthermore, landmark detection and association are performed by a human operator.

Drocourt et al. use in [175] a stereoscopic vision system equipped with a cone-shaped mirror and a camera that captures the reflected light, enabling an omnidirectional view of the scene. Vertical landmarks are extracted based on homogeneous regions on the cone. Based on the detection regions, a range and bearing measurement are computed, considering the distance and bearing-angle uncertainty. The pose is initially predicted using odometry information that shifts and inflates the refined vehicle pose subpaving at each new time step. Vertical landmarks in consecutive frames are matched by checking the intersection of the current measurement transformed by the predicted pose with the already mapped landmarks. Suppose inconsistent measurements lead to an empty set. The GOMNE algorithm increases the q in the q-relaxed intersection according to Subsection 2.2.5 until the resulting relative pose is not empty. However, the proposed method only works in closed laboratory environments and is not real-time capable.

Bethencourt and Jaulin propose in [176] v-SLAM approach with an RGBD camera (Kinect) using interval analysis. The main idea is to define initially large intervals for the transformation parameters for the relative pose between consecutive frames and then contract those intervals concurrently using common feature observations. Each observed feature defines constraints

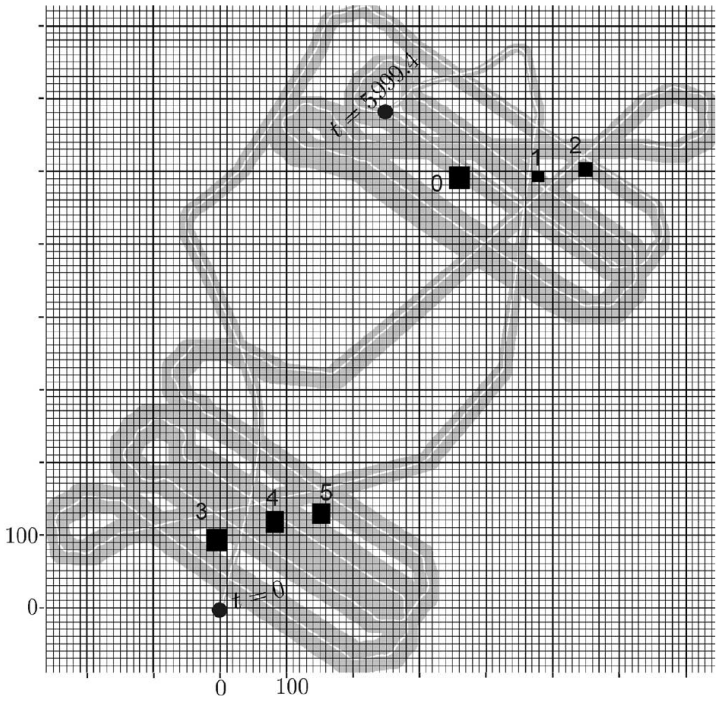


Figure B.5: Interval-based SLAM result. The interval approach provides a consistent envelope enclosing the submarine's trajectory, and six seamarks are detected and the map taking the interval uncertainty into account [174].

restricting the feasible set of relative pose parameters. To accelerate the contraction of the orientation parameters, the authors use an additional IMU that provides further constraints for the relative orientation. The ASIFT features are detected in the color image, and features in consecutive frames are matched using the descriptor. As the RGBD sensor also provides the depth information for the detected feature, each feature provides a simple transformation constraint that a forward-backward contractor implements. To deal with incorrect feature associations, a q-relaxed intersection is applied.

Vincke proposes in [177] an interval-based v-SLAM algorithm called Constraint Propagation SLAM (CP-SLAM) that uses wheel odometry and a monocular camera. The authors assume that the vehicle only moves on a flat surface. As a result, the robot pose can be parametrized by the two-dimensional position and the orientation. However, due to the scale ambiguity in monocular vision interval analysis makes a better parametrization of landmarks possible: While the observations ray and the robot pose needs to be considered, the depth of an observation can be defined by $[0, \infty]$ instead of the inverse depth as it is done in classical probabilistic approaches. As the observation ray is defined as a cone that goes from the camera projection center to infinity, the location of the landmark seen from different perspectives is directly obtained by the intersection. This innovative parametrization is only compatible since interval uncertainty is used. As image features, the authors use SURF [178]. Mapped landmarks are matched to detected features in two stages: First, the landmarks in the camera's FoV are projected into the image plane. Note that a landmark is defined by a box determined by the intersection of multiple views. However, if the perspective change has not been large enough, the depth estimate of the landmark will be comparatively large, resulting in a large 3D box. The projection region is defined as the search region. Second, this search region is used to

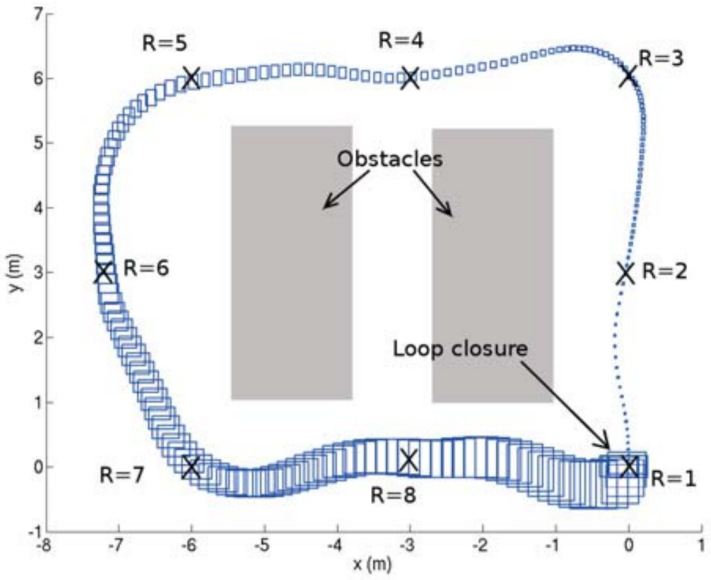


Figure B.6: CP-SLAM results. The uncertainty accumulates until loop closure constraints are included [177].

find a feature with the most similar descriptor. The pose estimation is classically divided into two steps. First, the pose is updated using the odometry information. Second, the landmark observation constraints are considered in the correction step to further contract the pose estimate. This approach is also capable of detecting loops. Such loop closures are essential to reduce the uncertainty. Figure B.6 illustrates the experimental results of a small trajectory with a detected loop. Note that the uncertainty accumulates until the loop closure constraints reduce the uncertainty again.

Similar to [176], Mustafa et al. suggest an RGBD camera-based interval SLAM approach [179]. However, instead of relying on distinct features as landmarks, the authors extract planes by first segmenting the color image and then clustering the corresponding point cloud with a RANSAC approach to planes. An interval-based approach determines the plane parameters from the clustered points for the plane patch. To distinguish and robustly track the planar patches, SURF features [178] are extracted from the planar image patch. Plane patches are used as landmarks are matched across multiple frames using a bag-of-words approach [180] that exploits the set of feature descriptors. The matched planes constrain the robot's relative pose using a forward-backward contractor with SIVIA. While the approach works in small and well-defined environments, that approach is severely limited in the scalability of larger environments. Furthermore, the approach is not online capable.

All interval-based methods that we have considered up to now only operate in closed regions with many loop closure constraints. Simple visual odometry that does not consider loop closure is doomed to fail using intervals, as the observation uncertainty will continuously accumulate along the trajectory. Voges and Wagner experimentally show in [51, 140] that visual odometry continuously accumulates the observation uncertainty, making the interval estimate useless. Other global information is vital to bound the uncertainty accumulation. Voges and Wagner, for instance, cope with this problem by incorporating very accurate GPS information. While the GPS data is only available at low frequency, the authors implement

an interval visual-LiDAR odometry to estimate the vehicle motion at sensor frequency. To accomplish that, the authors detect and track Shi-Tomasi features [155] in the image from a monocular camera. By projecting the LiDAR points and considering the interval uncertainties, some features are further augmented by depth information. Tracking those features, with or without depth, the relative transformation between consecutive frames can be determined by contracting an initially large pose interval vector to a smaller pose estimate by applying a sequence of forward-backward contractors for each tracked feature. The authors tackle outliers by applying a q-relaxed intersection. However, setting the value for q is empirical and cannot be determined in a guaranteed way. Furthermore, the method yields pessimistic results and is not real-time capable.

B.3 Hybrid Approaches

In the previous sections, we saw that probabilistic approaches are more popular than interval-based solutions for v-SLAM and visual odometry. One of the main reasons for that is that the uncertainty estimation is often neglected. While MLE approaches provide impressive results, the uncertainty estimate is often corrupted by linearization errors and/or covariance scalings inserted to set the importance of defined measurements but do not reflect the uncertainty properly. Consequently, hybrid approaches may yield better results by combining the good parts of those two worlds. However, to the best of our knowledge, no hybrid works exist in the context of v-SLAM and visual odometry. With this work, we intend to fill this gap.

Bibliography

[1] S. B. (Destatis), "Verkehr 2021", https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Publikationen/Downloads-Verkehrsunfaelle/verkehrsunfaelle-jahr-2080700217004.pdf, [Online; accessed 27-April-2023], 2023.

[2] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges", *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1275–1313, Secondquarter 2019.

[3] M. Maurer, B. Lenz, J. C. Gerdes, and H. Winner, Eds., *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*. Springer Vieweg, 2015, pp. 9–37.

[4] M. Wörner, F. Schuster, F. Dölitzscher, C. G. Keller, M. Haueis, and K. Dietmayer, "Integrity for autonomous driving: A survey", in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, Apr. 2016, pp. 666–671.

[5] "Road vehicles – Functional safety – Part 1: Vocabulary", International Organization for Standardization, Geneva, CH, Standard, Dec. 2018.

[6] A. K. Verma, S. Ajit, and D. R. Karanki, "Uncertainty analysis in reliability/safety assessment", in *Reliability and Safety Engineering*. London: Springer London, 2016, pp. 457–491.

[7] M. Tossaint, J. Samson, F. Toran, J. Ventura-Traveset, M. Hernandez-Pajarez, J. Juan, J. Sanz, and P. Ramos-Bosch, "The stanford – esa integrity diagram: A new tool for the user domain sbas integrity assessment", *Navigation*, vol. 54, no. 2, pp. 153–162, 2007.

[8] M. King and J. Kay, *Radical Uncertainty: Decision-making for an unknowable future*. Hachette UK, 2020.

[9] K. F. Park and Z. Shapira, "Risk and uncertainty", in *The Palgrave Encyclopedia of Strategic Management*, M. Augier and D. J. Teece, Eds. London: Palgrave Macmillan UK, 2017, pp. 1–7.

[10] V. Kreinovich and S. Shary, "Interval methods for data fitting under uncertainty: A probabilistic treatment", *Reliable Computing*, vol. 23, pp. 105–140, Jul. 2016.

[11] R. Neuland, R. Maffei, L. Jaulin, E. Prestes, and M. Kolberg, "Improving the precision of auvs localization in a hybrid interval-probabilistic approach using a set-inversion strategy", *Unmanned Systems*, vol. 2, pp. 361–375, Oct. 2014.

[12] A. Ehambram, R. Voges, C. Brenner, and B. Wagner, "Interval-based visual-inertial lidar slam with anchoring poses", in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 7589–7596.

[13] D. Dubois and H. Prade, "Possibility theory", in *Encyclopedia of Complexity and Systems Science*, R. A. Meyers, Ed. New York, NY: Springer New York, 2009, pp. 6927–6939.

[14] D. Dubois and H. Prade, "Possibility theory and its applications: Where do we stand?", *Mathware and Soft Computing Magazine*, vol. 18, Jan. 2015.

[15] A. Neumaier, "Clouds, fuzzy sets, and probability intervals", *Reliable Computing*, vol. 10, no. 4, pp. 249–272, 2004.

[16] G. Nassreddine, F. Abdallah, and T. Denoeux, "A new method for state estimation of dynamic system based on dempster shafer theory", in *2009 International Conference on Advances in Computational Tools for Engineering Applications*, Jul. 2009, pp. 101–106.

[17] G. Nassreddine, F. Abdallah, and T. Denoux, "State estimation using interval analysis and belief-function theory: Application to dynamic vehicle localization", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 5, pp. 1205–1218, Oct. 2010.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[19] A. A. Borovkov, "Random variables and distribution functions", in *Probability Theory*. London: Springer London, 2013, pp. 31–63.

[20] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam", *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, Dec. 2010.

[21] H. Strasdat, J. Montiel, and A. J. Davison, "Visual slam: Why filter?", *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012.

[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[23] M. L. A. Lourakis and A. A. Argyros, "Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?", in *Tenth IEEE Inter-*

national Conference on Computer Vision (ICCV'05), vol. 2, Oct. 2005, pp. 1526–1531.

[24] W. Förstner and B. P. Wrobel, *Photogrammetric computer vision*. Springer, 2016.

[25] K. MacTavish and T. D. Barfoot, "At all costs: A comparison of robust cost functions for camera correspondence outliers", in *2015 12th Conference on Computer and Robot Vision*, Jun. 2015, pp. 62–69.

[26] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras", *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[27] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam", in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 834–849.

[28] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras", in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 2100–2106.

[29] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization", in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 3607–3613.

[30] S. Geman, D. E. McClure, and D. Geman, "A nonlinear filter for film restoration and other problems in image processing", *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 4, pp. 281–289, 1992.

[31] J. R. Choike, "The pentagram and the discovery of an irrational number", *The Two-Year College Mathematics Journal*, vol. 11, no. 5, pp. 312–316, 1980.

[32] Archimedes, "Measurement of a circle", in *The Works of Archimedes: Edited in Modern Notation with Introductory Chapters*, T. L. Heath, Ed., ser. Cambridge Library Collection - Mathematics. Cambridge University Press, 2009, pp. 91–98.

[33] R. Moore, *Interval Analysis*, ser. Prentice-Hall series in automatic computation. Prentice-Hall, 1966.

[34] A. Neumaier, *Interval Methods for Systems of Equations*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1991.

[35] J. Rohn, "Systems of linear interval equations", *Linear Algebra and its Applications*, vol. 126, pp. 39–78, 1989.

[36] N. Apostolatos and U. Kulisch, "Grundlagen einer maschinenintervallarithmetik", *Computing*, vol. 2, no. 2, pp. 89–104, 1967.

[37] U. W. Kulisch and W. L. Miranker, "The arithmetic of the digital computer: A new approach", *SIAM Review*, vol. 28, no. 1, pp. 1–40, 1986.

[38] F. Schweppe, "Recursive state estimation: Unknown but bounded errors and system inputs", *IEEE Transactions on Automatic Control*, vol. 13, no. 1, pp. 22–28, Feb. 1968.

[39] E. Fogel and Y. Huang, "On the value of information in system identification—bounded noise case", *Automatica*, vol. 18, no. 2, pp. 229–238, 1982.

[40] K. Nickel, "Bericht über neue Karlsruher Ergebnisse bei der Fehlererfassung von numerischen Prozessen", German, *Apl. Mat.*, vol. 13, pp. 168–173, 1968.

[41] K. Nickel, "Über die Notwendigkeit einer Fehlerschranken-Arithmetik für Rechenautomaten", German, *Numer. Math.*, vol. 9, pp. 69–97, 1966.

[42] E. R. Hansen, "Global optimization using interval analysis: The one-dimensional case", *Journal of Optimization Theory and Applications*, vol. 29, no. 3, pp. 331–344, 1979.

[43] R. B. Kearfott, "Optimization", in *Rigorous Global Search: Continuous Problems*. Boston, MA: Springer US, 1996, pp. 169–208.

[44] G. Trombettoni, I. Araya, B. Neveu, and G. Chabert, "Inner regions and interval linearizations for global optimization", vol. 1, Aug. 2011.

[45] I. Araya, G. Trombettoni, B. Neveu, and G. Chabert, "Upper bounding in inner regions for global optimization under inequality constraints", *Journal of Global Optimization*, vol. 60, no. 2, pp. 145–164, 2014.

[46] E. Walter and H. Piet-Lahanier, "Exact recursive polyhedral description of the feasible parameter set for bounded-error models", *IEEE Transactions on Automatic Control*, vol. 34, no. 8, pp. 911–915, Aug. 1989.

[47] D. Y. Rokityanskiy and S. M. Veres, "Application of ellipsoidal estimation to satellite control design", *Mathematical and Computer Modelling of Dynamical Systems*, vol. 11, no. 2, pp. 239–249, 2005.

[48] C. Combastel, "A state bounding observer for uncertain non-linear continuous-time systems based on zonotopes", in *Proceedings of the 44th IEEE Conference on Decision and Control*, Dec. 2005, pp. 7228–7234.

[49] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer London Ltd, Aug. 2001, p. 398.

[50] S. Rohou, L. Jaulin, L. Mihaylova, F. Bars, and S. Veres, *Reliable Robot Localization: A Constraint-Programming Approach Over Dynamical Systems*. Oct. 2019.

[51] R. Voges, "Bounded-error visual-lidar odometry on mobile robots under consideration of spatiotemporal uncertainties", PhD thesis, Hannover: Institutionelles Repositorium der Leibniz Universität Hannover, 2020.

[52] G. Chabert. (2022). Ibex, a c++ library for constraint processing over real numbers, [Online]. Available: http://www.ibex-lib.org (visited on 07/12/2022).

[53] J. Sliwka, F. Le Bars, O. Reynet, and L. Jaulin, "Using interval methods in the context of robust localization of underwater robots", in *2011 Annual Meeting of the North American Fuzzy Information Processing Society*, Mar. 2011, pp. 1–6.

[54] B. Desrochers, "Simultaneous Localization and Mapping in Unstructured Environments", Theses, Université Bretagne Loire, May 2018.

[55] L. Jaulin, "Robust set-membership state estimation; application to underwater robotics", *Automatica*, vol. 45, no. 1, pp. 202–206, 2009.

[56] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[57] R. Voges and B. Wagner, "Set-Membership Extrinsic Calibration of a 3D LiDAR and a Camera", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, On-Demand Conference, Oct. 2020.

[58] "Lidar laser zur detektierung von hindernissen hdl-64e", [Online; accessed March 22, 2023], 2023.

[59] N. Li, C. P. Ho, J. Xue, L. W. Lim, G. Chen, Y. H. Fu, and L. Y. T. Lee, "A progress review on solid-state lidar and nanophotonics-based lidar sensors", *Laser & Photonics Reviews*, vol. 16, no. 11, pp. 210–511, 2022.

[60] "Vista-p90", [Online; accessed March 22, 2023], 2023.

[61] D. V. Nam and K. Gon-Woo, "Solid-state lidar based-slam: A concise review and application", in *2021 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jan. 2021, pp. 302–305.

[62] B. Zhou, D. Xie, S. Chen, H. Mo, C. Li, and Q. Li, "Comparative analysis of slam algorithms for mechanical lidar and solid-state lidar", *IEEE Sensors Journal*, vol. 23, no. 5, pp. 5325–5338, Mar. 2023.

[63] "Introduction", in *GNSS — Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Vienna: Springer Vienna, 2008, pp. 1–12.

[64] "Satellite signals", in *GNSS — Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Vienna: Springer Vienna, 2008, pp. 55–104.

[65] PROJ contributors, *PROJ coordinate transformation software library*, Open Source Geospatial Foundation, 2023.

[66] "Gps", in *GNSS — Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Vienna: Springer Vienna, 2008, pp. 309–340.

[67] "Reference systems", in *GNSS — Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Vienna: Springer Vienna, 2008, pp. 13–25.

[68] "Observables", in *GNSS — Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Vienna: Springer Vienna, 2008, pp. 105–160.

[69] G. Gröger and L. Plümer, "Citygml – interoperable semantic 3d city models", *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 71, pp. 12–33, 2012.

[70] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps", *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

[71] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation", *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, Jul. 2003.

[72] J.-S. Gutmann and D. Fox, "An experimental comparison of localization methods continued", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, Sep. 2002, pp. 454–459.

[73] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons", *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[74] L. Iocchi, D. Mastrantuono, and D. Nardi, "A probabilistic approach to hough localization", in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 4, May 2001, pp. 4250–4255.

[75] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures", *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.

[76] L. Teslić, I. Škrjanc, and G. Klančar, "Ekf-based localization of a wheeled mobile robot in structured environments", *Journal of Intelligent & Robotic Systems*, vol. 62, no. 2, pp. 187–203, May 2011.

[77] T. T. Hoang, P. M. Duong, N. T. T. Van, D. A. Viet, and T. Q. Vinh, "Multi-sensor perceptual system for mobile robot and sensor fusion-based localization", in

2012 International Conference on Control, Automation and Information Sciences (ICCAIS), Nov. 2012, pp. 259–264.

[78] C. Landsiedel and D. Wollherr, "Global localization of 3d point clouds in building outline maps of urban outdoor environments", *International journal of intelligent robotics and applications*, vol. 1, no. 4, pp. 429–441, 2017.

[79] P. Jensfelt and S. Kristensen, "Active global localization for a mobile robot using multiple hypothesis tracking", *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 748–760, Oct. 2001.

[80] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar", in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, Mar. 1985, pp. 116–121.

[81] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids", in *Proceedings of the national conference on artificial intelligence*, 1996, pp. 896–901.

[82] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments", *Journal of artificial intelligence research*, vol. 11, pp. 391–427, 1999.

[83] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots", in *Proceedings 1999 IEEE International Conference on Robotics and Automation*, vol. 2, May 1999, pp. 1322–1328.

[84] D. Fox, "Kld-sampling: Adaptive particle filters", in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14, MIT Press, 2001.

[85] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots", *Artificial Intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[86] M. Hentschel and B. Wagner, "Autonomous robot navigation based on openstreetmap geodata", in *13th International IEEE Conference on Intelligent Transportation Systems*, Sep. 2010, pp. 1645–1650.

[87] G. Floros, B. van der Zander, and B. Leibe, "Openstreetslam: Global vehicle localization using openstreetmaps", in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 1054–1059.

[88] P. Ruchti, B. Steder, M. Ruhnke, and W. Burgard, "Localization on openstreetmap data using a 3d laser scanner", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 5260–5265.

[89] F. Yan, O. Vysotska, and C. Stachniss, "Global localization on openstreetmap using 4-bit semantic descriptors", in *2019 European Conference on Mobile Robots (ECMR)*, Sep. 2019, pp. 1–7.

[90] Y. Chen, W. Chen, L. Zhu, Z. Su, X. Zhou, Y. Guan, and G. Liu, "A study of sensor-fusion mechanism for mobile robot global localization", *Robotica*, vol. 37, no. 11, pp. 1835–1849, 2019.

[91] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss, "Range image-based lidar localization for autonomous vehicles", in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 5802–5808.

[92] Q.-b. Zhang, P. Wang, and Z.-h. Chen, "An improved particle filter for mobile robot localization based on particle swarm optimization", *Expert Systems with Applications*, vol. 135, pp. 181–193, 2019.

[93] X. Chen, T. Läbe, L. Nardi, J. Behley, and C. Stachniss, "Learning an overlap-based observation model for 3d lidar localization", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 4602–4608.

[94] M. Zhou, X. Chen, N. Samano, C. Stachniss, and A. Calway, "Efficient localisation using images and openstreetmaps", in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 5507–5513.

[95] R. Neuland, M. Mantelli, B. Hummes, L. Jaulin, R. Maffei, E. Prestes, and M. Kolberg, "Robust hybrid interval-probabilistic approach for the kidnapped robot problem", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 29, no. 02, pp. 313–331, 2021.

[96] C. Olson, "Probabilistic self-localization for mobile robots", *IEEE Transactions on Robotics and Automation*, vol. 16, no. 1, pp. 55–66, Feb. 2000.

[97] S. Shan, "A levenberg-marquardt method for large-scale bound-constrained nonlinear least-squares", eng, PhD thesis, 2008.

[98] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes", in *Sensor Fusion IV: Control Paradigms and Data Structures*, P. S. Schenker, Ed., International Society for Optics and Photonics, vol. 1611, SPIE, 1992, pp. 586–606.

[99] S. Pang, D. Kent, X. Cai, H. Al-Qassab, D. Morris, and H. Radha, "3d scan registration based localization for autonomous vehicles - a comparison of ndt and icp under realistic conditions", in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Aug. 2018, pp. 1–5.

[100] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp.", in *Robotics: science and systems*, Seattle, WA, vol. 2, 2009, p. 435.

[101] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "Kiss-icp: In defense of point-to-point icp – simple, accurate, and robust registration if done the right way", 2022.

[102] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "Ct-icp: Real-time elastic lidar odometry with loop closure", in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5580–5586.

[103] O. Vysotska and C. Stachniss, "Exploiting building information from publicly available maps in graph-based slam", in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 4511–4516.

[104] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, "Robust lidar-based localization in architectural floor plans", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 3318–3324.

[105] D. Wilbers, C. Merfels, and C. Stachniss, "Localization with sliding window factor graphs on third-party maps for automated driving", in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5951–5957.

[106] S. Ratz, M. Dymczyk, R. Siegwart, and R. Dubé, "Oneshot global localization: Instant lidar-visual pose estimation", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 5415–5421.

[107] Y. Cho, G. Kim, S. Lee, and J.-H. Ryu, "Openstreetmap-based lidar global localization in urban environment without a prior lidar map", *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4999–5006, Apr. 2022.

[108] M. Kieffer, L. Jaulin, É. Walter, and D. Meizel, "Robust autonomous robot localization using interval analysis", *Reliable Computing*, vol. 6, no. 3, pp. 337–362, Aug. 2000.

[109] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel, "Guaranteed mobile robot tracking using interval analysis", in *MISC'99, Workshop on Application of Interval Analysis to System and Control*, Girona, Spain, Feb. 1999, pp. 347–360.

[110] E. Seignez, M. Kieffer, A. Lambert, E. Walter, and T. Maurin, "Experimental vehicle localization by bounded-error state estimation using interval analysis", in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug. 2005, pp. 1084–1089.

[111] I. A. R. Ashokaraj, P. M. G. Silson, A. Tsourdos, and B. A. White, "Robust sensor-based navigation for mobile robots", *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 3, pp. 551–556, 2009.

[112] E. Seignez, M. Kieffer, A. Lambert, E. Walter, and T. Maurin, "Real-time bounded-error state estimation for vehicle tracking", *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 34–48, Jan. 2009.

[113] A. Gning and P. Bonnifait, "Guaranteed dynamic localization using constraints propagation techniques on real intervals", in *IEEE International Conference on Robotics and Automation*, vol. 2, Apr. 2004, pp. 1499–1504.

[114] A. Gning and P. Bonnifait, "Dynamic vehicle localization using constraints propagation techniques on intervals a comparison with kalman filtering", in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 4144–4149.

[115] A. Gning and P. Bonnifait, "Constraints propagation techniques on intervals for a guaranteed localization using redundant data", *Automatica*, vol. 42, no. 7, pp. 1167–1175, 2006.

[116] A. Lambert, D. Gruyer, B. Vincke, and E. Seignez, "Consistent outdoor vehicle localization by bounded-error state estimation", in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 1211–1216.

[117] A. Clérentin, M. Delafosse, L. Delahoche, B. Marhic, and A.-M. Jolly-Desodt, "Uncertainty and imprecision modeling for the mobile robot localization problem", *Autonomous Robots*, vol. 24, no. 3, pp. 267–283, 2008.

[118] J. Sliwka, L. Jaulin, M. Ceberio, and V. Kreinovich, "Processing interval sensor data in the presence of outliers, with potential applications to localizing underwater robots", in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 2011, pp. 2330–2337.

[119] M. Langerwisch and B. Wagner, "Guaranteed mobile robot tracking using robust interval constraint propagation", in *Intelligent Robotics and Applications*, C.-Y. Su, S. Rakheja, and H. Liu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 354–365.

[120] R. Guyonneau, S. Lagrange, L. Hardouin, and P. Lucidarme, "Guaranteed interval analysis localization for mobile robots", *Advanced Robotics*, vol. 28, no. 16, pp. 1067–1077, Jul. 2014.

[121] B. Desrochers, S. Lacroix, and L. Jaulin, "Set-membership approach to the kidnapped robot problem", in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3715–3720.

[122] I.-F. Kenmogne, V. Drevelle, and E. Marchand, "Image-based uav localization using interval methods", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 5285–5291.

[123] U. D. Hanebeck and J. Horn, "A new estimator for mixed stochastic and set theoretic uncertainty models applied to mobile robot localization", in *Proceedings 1999 IEEE International Conference on Robotics and Automation*, vol. 2, May 1999, pp. 1335–1340.

[124] L. Jaulin, "Set-membership localization with probabilistic errors", *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 489–495, 2011.

[125] I. Ashokaraj, A. Tsourdos, P. Silson, and B. White, "Sensor based robot localisation and navigation: Using interval analysis and extended kalman filter", in *2004 5th Asian Control Conference*, vol. 2, Jul. 2004, pp. 1086–1093.

[126] M. Louédec and L. Jaulin, "Interval extended kalman filter—application to underwater localization and control", *Algorithms*, vol. 14, no. 5, 2021.

[127] F. Abdallah, A. Gning, and P. Bonnifait, "Box particle filtering for nonlinear state estimation using interval analysis", *Automatica*, vol. 44, no. 3, pp. 807–815, 2008.

[128] P. Wang, L. Mihaylova, P. Bonnifait, P. Xu, and J. Jiang, "Feature-refined box particle filtering for autonomous vehicle localisation with openstreetmap", *Engineering Applications of Artificial Intelligence*, vol. 105, p. 104 445, 2021.

[129] R. Neuland, J. Nicola, R. Maffei, L. Jaulin, E. Prestes, and M. Kolberg, "Hybridization of monte carlo and set-membership methods for the global localization of underwater robots", in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 199–204.

[130] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf", in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2564–2571.

[131] R. Mur-Artal and J. D. Tardós, "Fast relocalisation and loop closing in keyframe-based slam", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 846–853.

[132] R. Mur-Artal and J. D. Tardós, "Orb-slam: Tracking and mapping recognizable", in *Workshop on Multi View Geometry in Robotics (MVIGRO)-RSS*, 2014.

[133] R. Mur-Artal and J. D. Tardós, "Probabilistic semi-dense mapping from highly accurate feature-based monocular slam.", in *Robotics: Science and Systems*, Rome, vol. 2015, 2015.

[134] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system", *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[135] OpenMP Architecture Review Board, "OpenMP application program interface version 4.5", 2015.

[136] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", in *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, Vancouver, Canada, Aug. 1981, pp. 674–679.

[137] A. Ehambram, R. Voges, and B. Wagner, "Stereo-visual-lidar sensor fusion using set-membership methods", in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Aug. 2021, pp. 1132–1139.

[138] L. Jaulin, M. Kieffer, E. Walter, and D. Meizel, "Guaranteed robust nonlinear estimation with application to robot localization", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 4, pp. 374–381, Nov. 2002.

[139] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces", in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nov. 2007, pp. 225–234.

[140] R. Voges and B. Wagner, "Interval-based visual-lidar sensor fusion", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1304–1311, Apr. 2021.

[141] A. Ehambram, L. Jaulin, and B. Wagner, "Interval-based global localization in building maps", *SWIM 2022*, 2022.

[142] O. Wulf, K. Arras, H. Christensen, and B. Wagner, "2d mapping of cluttered indoor environments by means of 3d perception", in *IEEE International Conference on Robotics and Automation*, vol. 4, Apr. 2004, pp. 4204–4209.

[143] A. Ehambram, L. Jaulin, and B. Wagner, "Hybrid interval-probabilistic localization in building maps", *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7059–7066, Jul. 2022.

[144] S. Rohou, B. Desrochers, and L. Jaulin, "Set-membership state estimation by solving data association", in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 4393–4399.

[145] L. Jaulin and S. Bazeille, "Image shape extraction using interval methods", *IFAC Proceedings Volumes*, vol. 42, no. 10, pp. 378–383, 2009, 15th IFAC Symposium on System Identification.

[146] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset", en, *International journal of robotic research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[147] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters", *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[148] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet ++: Fast and accurate lidar semantic segmentation", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 4213–4220.

[149] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[150] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[151] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry", in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, Jun. 2004, pp. I–I.

[152] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections", *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[153] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]", *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[154] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[155] Jianbo Shi and Tomasi, "Good features to track", in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 1994, pp. 593–600.

[156] L. M. Paz, P. PiniÉs, J. D. TardÓs, and J. Neira, "Large-scale 6-dof slam with stereo-in-hand", *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, Oct. 2008.

[157] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem", *Proceedings of the National Conference on Artificial Intelligence*, Nov. 2002.

[158] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, *et al.*, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges", in *IJCAI*, 2003, pp. 1151–1156.

[159] R. Sim, P. Elinas, M. Griffin, J. J. Little, *et al.*, "Vision-based slam using the rao-blackwellised particle filter", in *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, vol. 14, 2005, pp. 9–16.

[160] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[161] E.-y. Wu, G.-y. Li, Z.-y. Xiang, and J.-l. Liu, "Stereo vision based slam using rao-blackwellised particle filter", *Journal of Zhejiang University-SCIENCE A*, vol. 9, no. 4, pp. 500–509, Apr. 2008.

[162] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection", in *Computer Vision - ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[163] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks", *The International Journal of Robotics Research*, vol. 21, no. 8, pp. 735–758, 2002.

[164] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam", in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2352–2359.

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

[166] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras", in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 1935–1942.

[167] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2174–2181.

[168] W. Shao, S. Vijayarangan, C. Li, and G. Kantor, "Stereo visual inertial lidar simultaneous localization and mapping", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 370–377.

[169] G. D. Tipaldi, G. Grisetti, and W. Burgard, "Approximate covariance estimation in graphical approaches to slam", in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2007, pp. 3460–3465.

[170] V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik, "Fast covariance recovery in incremental nonlinear least square solvers", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4636–4643.

[171] J. Ortiz, T. Evans, and A. J. Davison, "A visual introduction to gaussian belief propagation", 2021.

[172] J. Ortiz, M. Pupilli, S. Leutenegger, and A. J. Davison, "Bundle adjustment on a graph processor", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[173] A. J. Davison and J. Ortiz, "Futuremapping 2: Gaussian belief propagation for spatial ai", 2019.

[174] L. Jaulin, "A nonlinear set membership approach for the localization and map building of underwater robots", *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 88–98, Feb. 2009.

[175] C. Drocourt, L. Delahoche, E. Brassart, B. Marhic, and A. Clérentin, "Incremental construction of the robot's environmental map using interval analysis", in *Global Optimization and Constraint Satisfaction*, C. Jermann, A. Neumaier, and D. Sam, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 127–141.

[176] A. Bethencourt and L. Jaulin, "3d reconstruction using interval methods on the kinect device coupled with an imu", *International Journal of Advanced Robotic Systems*, vol. 10, no. 2, p. 93, 2013.

[177] B. Vincke, A. Lambert, and A. Elouardi, "Guaranteed simultaneous localization and mapping algorithm using interval analysis", in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, Dec. 2014, pp. 1409–1414.

[178] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features", in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[179] M. Mustafa, A. Stancu, N. Delanoue, and E. Codres, "Guaranteed slam—an interval approach", *Robotics and Autonomous Systems*, vol. 100, pp. 160–170, 2018.

[180] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences", *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

Publications

Publications as main author

A. Ehambram, P. Hemme, and B. Wagner, "An approach to marker detection in IR- and rgb-images for an augmented reality marker", in *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2019 - Volume 2, Prague, Czech Republic, July 29-31, 2019*, O. Gusikhin, K. Madani, and J. Zaytoon, Eds., SciTePress, 2019, pp. 190–197.

A. Ehambram, H. Homann, S. P. Kleinschmidt, T. Ritter, N. Fischer, and B. Wagner, "Hierarchic single cluster graph partitioning: A sequential place recognition method", in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2020, pp. 1398–1405.

A. Ehambram, R. Voges, and B. Wagner, "Stereo-visual-lidar sensor fusion using set-membership methods", in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Aug. 2021, pp. 1132–1139.

A. Ehambram, R. Voges, C. Brenner, and B. Wagner, "Interval-based visual-inertial lidar slam with anchoring poses", in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 7589–7596.

A. Ehambram, L. Jaulin, and B. Wagner, "Hybrid interval-probabilistic localization in building maps", *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7059–7066, Jul. 2022.

A. Ehambram, L. Jaulin, and B. Wagner, "Interval-based global localization in building maps", *SWIM 2022*, 2022.

Publications as co-author

P. Venet, K. Safronov, A. Ehambram, S. Wagner, J. Bock, and U. E. Zimmermann, "Application of ontologies for semantic scene segmentation and object recognition", in *ISR 2020; 52th International Symposium on Robotics*, Dec. 2020, pp. 1–7.

Y. Jiang, A. Ehambram, and B. Wagner, "Interval-based robot localization with uncertainty evaluation", in *Proceedings of the 19th International Conference on Informatics in Control, Automation and Robotics*, INSTICC, vol. 1, SciTePress, 2022, pp. 296–303.

About the Author

Personal details

Name	Aaronkumar Ehambram
Date of birth	Januray 27th, 1996
Place of birth	Rinteln, Germany

Education

2002 – 2014	General school education Abitur at Gymnasium Ernestinum Rinteln
2014 – 2019	Studies in Electrical Engineering and Information Technology at Leibniz University Hannover B. Sc. and M. Sc. degrees

Scientific work

January 2020 – Present	Research associate at Leibniz University Hannover Institute of Systems Engineering – Real Time Systems Group DFG Research Training Group 2159 (i.c.sens)