Bounded-Error Visual-LiDAR Odometry on Mobile Robots Under Consideration of Spatiotemporal Uncertainties

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. Raphael Voges
geboren am 08.11.1992 in Hildesheim

2020

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

Tag der Promotion: 17. Juni 2020

Acknowledgements (mostly in German)

Die vorliegende Dissertation entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fachgebiet Echtzeitsysteme (RTS) des Instituts für Systems Engineering der Leibniz Universität Hannover. Des Weiteren war ich wissenschaftlicher Mitarbeiter des von der Deutschen Forschergemeinschaft (DFG) finanzierten Graduiertenkollegs i.c.sens [RTG 2159].

Ich möchte mich insbesondere bei meinem Doktorvater Prof. Dr.-Ing. Bernardo Wagner für die anhaltende Unterstützung, die exzellente Betreuung und die ständige Motivation bedanken.

Moreover, I am extremely grateful to Prof. Luc Jaulin for his hospitality in Brest, the numerous discussions surrounding this work and for his willingness to act as the co-examiner of my dissertation.

Weiterhin möchte ich meinen Kolleginnen und Kollegen des RTS und ZDT für die angenehme Zusammenarbeit und die inspirierenden Diskussionen danken. Im Speziellen bin ich Aaronkumar Ehambram, der wertvolle Hinweise zur finalen Version dieser Dissertation lieferte, dankbar.

In addition, I would like to thank Simon Rohou for the fruitful collaboration and the joint activities during my stay in Brest.

Besides, I am grateful to the members of the Research Training Group i.c.sens for the joint measurement campaigns as well as the constant interchange of new ideas during the frequent Jour Fixes and the Supervisors' Days.

Schließlich möchte ich meinen Eltern, meinem Bruder und meinen Freunden für die bedingungslose Unterstützung auch abseits des beruflichen Umfelds und das in mich gesetzte Vertrauen danken. Der größte Dank gebührt meiner Partnerin Julia, für das auch in anstrengenden Phasen meiner Arbeit entgegengebrachte Verständnis, für die ständige Motivation und dafür, dass sie mir stets den Rücken freigehalten hat.

Raphael Voges

Juni 2020

Kurzfassung

Mit dem Aufkommen des autonomen Fahrens wird die Lokalisierung mobiler Roboter, insbesondere auch ohne GNSS-Informationen, immer wichtiger. Dabei muss sichergestellt werden, dass die Lokalisierung robust funktioniert und rechtzeitig Warnungen ausgegeben werden, falls die Posenschätzung zu unsicher wird, um einen sicheren Betrieb des Systems zu gewährleisten. Zur Erfüllung dieser Anforderungen benötigen die autonomen Systeme zuverlässige und vertrauenswürdige Informationen über ihre Umgebung. Um die Zuverlässigkeit und Integrität dieser Informationen zu verbessern und robust gegenüber Sensorausfällen zu sein, sollten Informationen von mehreren Sensoren fusioniert werden. Dies erfordert jedoch, dass bestimmte Eigenschaften zwischen den Sensoren (z.B. die Transformation zwischen den Sensorkoordinatensystemen) bekannt sind. Allerdings sind weder die tatsächlichen Sensormessungen fehlerfrei, noch können diese Eigenschaften zwischen den Sensoren fehlerfrei bestimmt werden. Daher müssen diese Fehlerquellen während der Sensorfusion entsprechend modelliert werden.

Um autonome Fahrzeuge ohne GNSS-Informationen in 3D zu lokalisieren, wird in dieser Arbeit ein Dead-Reckoning-Ansatz vorgestellt, der Informationen von einer Kamera, einem Laserscanner und einer IMU fusioniert. Dafür werden zunächst neue Sensorfehlermodelle für jeden einzelnen Sensor aufgestellt. Hierbei werden die Sensorfehler als unbekannt, aber begrenzt angenommen. Dies erfordert, dass Grenzen (d. h. Intervalle), die von den tatsächlichen Fehlern nicht überschritten werden, bekannt sind. Weitere Annahmen sind nicht erforderlich. Insbesondere muss die Fehlerverteilung innerhalb der Intervalle nicht bekannt sein, was eine häufig nicht beachtete Annahme etablierter Ansätze ist. Außerdem sind Fehlermodelle, die auf der Intervallarithmetik beruhen, mit unbekannten systematischen Fehlern vereinbar und eignen sich, um Ergebnisse garantieren zu können. Des Weiteren werden in dieser Arbeit neuartige Ansätze für die raum-zeitliche Kalibrierung zwischen Kamera, Laserscanner und IMU vorgestellt, die die eingeführten Fehlermodelle verwenden, um nicht nur die Kalibrierparameter, sondern auch die zugehörigen Unsicherheiten zu bestimmen. Zuletzt wird ein neues Verfahren zur garantierten Informationsfusion aller Sensoren entwickelt. Hierbei werden sowohl die Fehler der einzelnen Sensoren als auch die Ungenauigkeiten der Kalibrierung zwischen den Sensoren berücksichtigt. Anschließend werden die fusionierten Informationen genutzt, um einen mobilen Roboter inkrementell in einem lokalen Koordinatensystem zu lokalisieren.

Bei der Evaluation stellt sich unter Nutzung sowohl simulierter als auch realer Daten heraus, dass alle vorgestellten Verfahren korrekte Ergebnisse garantieren, solange die angenommenen Fehlergrenzen zutreffend sind. Obwohl die intervallbasierten Ansätze den “worst case”, also die maximalen Sensorfehler, berücksichtigen, sind die Ergebnisse hinreichend genau. Insbesondere können Situationen, in denen stochastische Verfahren Ergebnisse berechnen, die erheblich von der tatsächlichen Lösung abweichen, identifiziert werden.

Schlagworte:
Autonomes Fahren, Lokalisierung, Koppelnavigation, Sensorfusion, Informationsfusion, Unsicherheitsmodellierung, Sensorfehlermodelle, Intervallarithmetik

Abstract

With the advent of autonomous driving, the localization of mobile robots, especially without GNSS information, is becoming increasingly important. It must be ensured that the localization works robustly and timely warnings are provided if the pose estimates are too uncertain to assure a safe operation of the system. To meet these requirements, autonomous systems require reliable and trustworthy information about their environment. To improve the reliability and the integrity of information, and to be robust with respect to sensor failures, information from multiple sensors should be fused. However, this requires inter-sensor properties (e.g. the transformation between sensor coordinate systems) to be known. Naturally, neither the actual sensor measurements nor the inter-sensor properties can be determined without errors, and thus must be modeled accordingly during sensor fusion.

To localize autonomous vehicles without GNSS information in 3D, this work introduces a dead reckoning approach relying on information from a camera, a laser scanner and an IMU. First, novel error models for the individual sensors are introduced. Here, the errors are assumed to be unknown but bounded, which requires bounds (i.e. intervals) that are not exceeded by the actual sensor errors to be known. However, no further assumptions are required. In particular, the error distribution within the bounds does not need to be known, which is a frequently overlooked assumption of established approaches. Furthermore, interval-based error models are compatible with unknown systematic errors and can be used to guarantee results. Second, to determine the inter-sensor properties and the corresponding uncertainties, this thesis presents new approaches for the spatiotemporal calibration between camera, laser scanner and IMU that employ the proposed error models. Third, an innovative method that considers both sensor and inter-sensor errors for guaranteed sensor fusion is proposed. The fused information is subsequently used to perform interval-based dead reckoning of a mobile robot.

To evaluate the developed methods, both simulated and real data are analyzed. It becomes evident that all proposed approaches are guaranteed to enclose the true solution if the sensor error bounds are correct. Moreover, although interval-based approaches consider the “worst case”, i.e. the maximum sensor errors, the results are reasonably accurate. In particular, it can be determined in which instances a state-of-the-art method computes a result that deviates significantly from the actual solution.

Keywords:
Autonomous Driving, Localization, Dead Reckoning, Sensor Fusion, Visual-LiDAR Odometry, Error Modeling, Sensor Error Models, Interval Analysis

Contents

Acronyms

CSP	Constraint Satisfaction Problem
DOF	Degrees Of Freedom
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
MRP	Modified Rodrigues Parameters
OpenCV	Open Source Computer Vision Library
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
SIVIA	Set Inversion Via Interval Analysis
SLAM	Simultaneous Localization and Mapping
ToF	Time of Flight

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
$x(\cdot)$	Trajectory (signal over time)
$[x](\cdot)$	Tube (set of trajectories)

Coordinate systems and transformations

C	Camera coordinate system
L	Laser scanner coordinate system
I	Inertial Measurement Unit (IMU) coordinate system
W	World coordinate system
$\mathbf{R}_B^A$	Rotation matrix describing the rotation between coordinate systems A and B
$\mathbf{T}_B^A$	Translation vector describing the shift between coordinate systems A and B
$\boldsymbol{\xi}_B^A = (\,\varphi_B^A \;\; \theta_B^A \;\; \psi_B^A\,)$	Euler angles describing the rotation between coordinate systems A and B
φ_B^A	Yaw angle
θ_B^A	Pitch angle
ψ_B^A	Roll angle

Camera coordinate systems

f, g	Image frames
f_i	Image frame i
k_i	Image keyframe i

C_{f_i}	Camera coordinate system during the acquistion of image i
$\mathbf{R}_g^f$	Rotation matrix describing the rotation between the acquistion of image frames f and k in the camera coordinate system C
$\mathbf{T}_g^f$	Translation vector describing the shift between the acquistion of image frames f and k in the camera coordinate system C

Coordinates

r, α, β	Spherical coordinates consisting of the distance r, polar angle α and azimuthal angle β
$\mathbf{X}_i^A = (\, x_i^A \;\; y_i^A \;\; z_i^A \,)^\intercal$	3D coordinates of the point i in the coordinate system A
$\tilde{\mathbf{X}}_i^A = (\, \tilde{x}_i^A \;\; \tilde{y}_i^A \;\; 1 \,)^\intercal$	Normalized 3D coordinates of the point i in the coordinate system A
$\mathbf{X}_i^f = (\, x_i^f \;\; y_i^f \;\; z_i^f \,)^\intercal$	3D coordinates of the point i in the camera coordinate system C during the acquistion of image frame f
$\tilde{\mathbf{X}}_i^f = (\, \tilde{x}_i^f \;\; \tilde{y}_i^f \;\; 1 \,)^\intercal$	Normalized 3D coordinates of the point i in the camera coordinate system C during the acquistion of image frame f

1 Introduction

736 billion kilometers. That is the distance covered by motor vehicles on German roads in 2018 according to the Federal Motor Transport Authority [1]. Nowadays, the goal is to automate these vehicles so that drivers eventually are no longer required. Consequently, mobile robots must be able to localize themselves in unknown environments with potentially tall buildings that may prevent information from a Global Navigation Satellite System (GNSS). In the absence of this global positioning information, the mobile robot must rely on its exteroceptive sensors (e.g. camera, laser scanner, etc.) and its proprioceptive sensors (e.g. IMU, wheel encoders, etc.) to localize itself. If additionally no map of the environment is available, the robot can only localize itself incrementally starting from an arbitrary pose that defines the origin of its local coordinate system. This procedure can be extended to the Simultaneous Localization and Mapping (SLAM) problem if a map is built simultaneously. Although much research has been conducted in this field, the question arises as to how the robustness, safety and integrity of each vehicle can be guaranteed if they are operated autonomously over the entire 736 billion kilometers [2].

In general, the terms robustness, safety and integrity involve many different aspects. One important aspect of robustness is the robustness with respect to the sensor errors. Naturally, sensors are required to gather information about the robot's motion and its environment. However, no sensor can achieve error-free measurements, and thus induces errors during the localization process. Consequently, these errors must first be identified, appropriately modeled, and subsequently propagated to guarantee the robustness of the localization results.

Although different models for sensor errors are available, in recent years the probabilistic error model has prevailed in mobile robotics. Often, sensor errors are assumed to follow a normal distribution. This has proven to be advantageous as following computations and error propagations are simple and straightforward. Consequently, the result of localization approaches is not only the robot's pose, but a stochastic distribution of possible poses.

The safety of a road vehicle is defined by the International Organization for Standardization (ISO), among other things, as the absence of unacceptable risk [3]. To quantify risk, the concept of integrity was introduced in the field of aviation. Among other things, it "includes the ability of a system to provide timely and valid warnings to the user (alerts) when the system must not be used for the intended operation" [4]. This definition can be directly transferred to the field of autonomous driving. For example, one unacceptable risk for an automated vehicle is a localization error that exceeds a predefined tolerance and that is not detected so that no

countermeasures (e.g. notifying the driver, slowing down, etc.) can be initiated. Consequently, the safety of an autonomous vehicle is directly related to its integrity.

To further increase the robustness of autonomous vehicles, it is advisable to combine information from different sensors to incrementally localize a mobile robot. Within this context, the incremental localization of a robot refers to the computation of the robot's pose in a local coordinate system that is established before the robot starts moving or even at an arbitrary point in time. This procedure, also known as *dead reckoning* or *odometry*, is an integral part of an autonomous vehicle that cannot be localized in a global coordinate system due to a lack of map information and tall buildings preventing GNSS information.

Three sensors commonly used for the incremental localization of mobile robots are the color camera, the Light Detection and Ranging (LiDAR) sensor (or laser scanner) and the IMU. These sensors are widely employed since they constitute different types of sensors that exhibit different advantages and disadvantages. For example, while the camera provides rich image features that can be re-identified at successive points in time and space, the depth of features (i.e. their distance from the camera) is hard to determine robustly [5]. In contrast, the laser scanner is able to measure accurate distance information, but does not provide features which can be re-identified easily. Unlike both these exteroceptive sensors, the IMU is useful for measuring the acceleration and rotational velocity of the robot. Due to these different characteristics, it is beneficial to fuse information from all three sensors. The process of computing the robot's incremental pose using data from both camera and laser scanner is usually denoted as *visual-LiDAR odometry*. Often, data from the IMU is also used without specifically referring to it in the name.

However, sensor fusion, which is the process of combining data from multiple sensors, introduces new challenges as inter-sensor properties have to be known. These so-called spatiotemporal calibration parameters include, for example, the spatial relation between sensor coordinate systems or the time offset between sensor clocks. Accordingly, we require methods to compute these parameters. Furthermore, the accuracy of fused information no longer depends on just the sensor errors, but also on the errors occurring during the calibration.

1.1 Problem statement and contributions

Although sensor errors can be modeled conveniently using a normal distribution, it is not the appropriate error model for each and every sensor. Usually, the true error distribution of a sensor deviates from a normal distribution, thus requiring a different stochastic error distribution. However, often this distribution is unknown and cannot be easily determined since it may differ from sensor to sensor and requires an extensive calibration to create statistically meaningful data. For example, a laser beam emitted by a laser scanner is not a perfect beam, but diverges. Now, when computing the error distribution of the resulting scan point, the energy distribution within the diverging beam must be taken into account. However, this energy distribution is generally unknown and may even change over time due to sensor deterioration.

Therefore, this work investigates an alternative error model based on interval analysis. Here, sensor errors are assumed to be unknown but bounded. Consequently, computations are carried out using the upper and lower bounds of the interval in which the true value resides.

Since the true error distribution does not need to be known, this bounded error model is consistent with any possible error distribution. However, bounds representing the maximum error must be known. Nonetheless, these bounds are often specified by the manufacturer (e.g. beam divergence of a laser scanner) or appear naturally due to physical fundamentals (e.g. discretization into camera pixels). Still, outliers violating these bounds can be taken into account, as we will show in this work.

Consequently, this fundamentally different way of modeling sensor errors requires new approaches for the fusion of sensor information and the dead reckoning of a mobile robot. Moreover, the need to comply with strict safety standards has been identified, which includes the ability to generate alerts, but not much research has been conducted in this field [6, 7]. For example, most authors evaluate the accuracy of their localization algorithms by computing the average deviation from the true position and neglect the maximum error their approach makes. However, for safety-critical systems the maximum error is of special interest since a single large error can cause the system to crash although it performs well on average.

To overcome these shortcomings, this works aims to make the following contributions:

- Investigation of possible error sources for camera, laser scanner and IMU, as well as subsequent development of new bounded error models for these sensors.
- Introduction of a novel so-called contractor $\mathcal{C}_{\text{offset}}$ to compute the time offset in the context of constraint programming over dynamical systems.
- Design of a new approach for the spatiotemporal calibration (extrinsic rotation and time offset) between camera and IMU under interval uncertainty.
- Development of a novel approach for the extrinsic calibration between camera and LiDAR under interval uncertainty.
- Development of an interval-based sensor fusion approach to combine information from camera and laser scanner. Here, the desired contribution of this work is to consider both the sensor error bounds and the uncertain extrinsic calibration parameters.
- Design of a new interval-based visual-LiDAR odometry approach for the dead reckoning of a mobile robot using the fused information and IMU data.
- Experimental evaluation of all depicted approaches using simulated and/or real data to assess whether interval-based error modeling can increase the robustness and integrity of autonomous vehicles.

1.1.1 Why interval analysis?

> *"I think it's much more interesting to live not knowing than to have answers which might be wrong. I have approximate answers and possible beliefs and different degrees of certainty about different things, but I'm not absolutely sure of anything and there are many things I don't know anything about."*
>
> Richard P. Feynman [8]

As in the quote of Richard P. Feynman, when measuring, for example, a distance with a laser scanner, we cannot determine the actual distance with absolute accuracy, but get an approximate answer. Intuitively, we as humans would state that we know the distance with an

accuracy of, for example, ± 1 cm. This is the idea of interval analysis. Rather than specifying an exact value or a stochastic distribution, it is assumed that measurements belong to an interval without making any statement about which value within the interval is most likely. Since all following computations are performed using intervals rather than scalar values, new arithmetic rules and algorithms have been defined [9]. Consequently, the output of these algorithms are also intervals that enclose the true result while simultaneously reflecting its uncertainty. The accumulation of all these algorithms and arithmetic rules is denoted as interval analysis or interval arithmetic. In the following, we list the advantages this alternative way of error modeling and error propagation offers over traditional stochastic error modeling for this work:

Unknown error distributions

As stated above, the true error distribution of sensors is often unknown. This is especially true if the measurement accuracy is significantly influenced by the manufacturing process and the manufacturer does not perform a specific calibration of each sensor. Furthermore, the error distribution can become unknown if the sensor is exposed to changing environmental conditions (e.g. temperature) that distort the measurement process. In these cases, stochastic error modeling is inappropriate since the error can be drastically under-estimated, thus resulting in a misleading outcome [10]. In contrast, interval analysis only requires maximum error bounds that are more convenient to obtain (e.g. maximum manufacturing inaccuracies or maximum allowed temperature). Besides, the measurement process of some sensors results in natural physical bounds (e.g. the encoder of a motor or the discretization into camera pixels). Nevertheless, if absolutely no information about the sensor error is available and the error bounds are chosen too large, subsequent computations are also inaccurate and cannot lead to significant knowledge gain.

Unknown systematic errors

Although unknown systematic errors are, strictly speaking, part of an unknown error distribution, we want to stress them especially. Generally, stochastic approaches assume zero-mean errors which means that the average error over a sufficient number of measurements equals zero. However, if, for example, a laser scanner systematically overestimates the true distance by 1 cm due to an imprecise calibration, this assumptions is violated. Consequently, an unknown systematic error cannot easily be taken into account using a stochastic error distribution, thus distorting the localization result. Gauss himself always precluded systematic errors since they were incompatible with his distribution model, nowadays referred to as the Gaussian (or normal) distribution [11]. In contrast, the bounded error model is consistent with unknown systematic errors since the only requirement is that the true value resides in the corresponding interval.

Error propagation

In general, different types of errors exhibit different error distributions. However, when employing stochastic error models, it is often hard to propagate these different errors consistently. Consequently, it is more convenient to use only the variances and not the full distribution for

the error propagation. However, this leads to a overly optimistic result since the errors are propagated quadratically, whereas Kutterer and Schön show that a linear propagation is more adequate for, for example, systematic errors [12]. Contrary, interval analysis offers a consistent error propagation that is guaranteed to not underestimate the resulting error [13]. In addition, in the case of failures or accidents of autonomous vehicles, the consistent error propagation of interval analysis allows to determine the cause, i.e. the violated assumption, and thus enables to unambiguously point out responsibilities (e.g. of the sensor manufacturer).

Guarantees

As stated above, autonomous cars are safety-critical systems for which the average error is less important than the maximum error. Therefore, interval analysis is well suited for these systems since it can be employed to compute guaranteed bounds enclosing the true result. Assuming the initial error bounds are correct, no possible solution is lost during the error propagation, leading to guaranteed results. Conversely, interval-based approaches also allow to dismiss infeasible solutions outside the interval, thus shrinking the search space for following computations. Consequently, outliers can be identified unambiguously. In contrast, stochastic approaches cannot provide reliable bounds since generally a nonzero probability remains for solutions far from the true result.

Determinism

Computations performed using interval analysis are deterministic. This means that given an input value, the result will remain the same no matter how often the computation is repeated. Especially for safety-critical systems that require certified algorithms, this property is important since no unpredictable results can occur. In contrast, stochastic approaches like the particle filter are random to a certain degree, thus leading to non-deterministic results.

Intuitive representation

As explained above, us humans naturally speak in intervals if we are uncertain about a specific value. If we are asked, for example, about the distance to the nearest bus stop, we might answer that it is 200 to 300 meters away. Consequently, intervals are an intuitive representation that can be easily understood. Moreover, instead of just specifying the value, intervals simultaneously indicate the value's accuracy. In contrast, a stochastic distribution is less intuitive and requires additional knowledge to depict the accuracy of a value.

Advantages over optimization algorithms

Common optimization algorithms (e.g. the Levenberg–Marquardt algorithm [14]), which are often employed to find the optimal parameters of a nonlinear function, require the function to be linearized. This introduces linearization errors since the optimal linearization point is unknown. Moreover, these optimization algorithms can converge to a local optimum and not the global optimum if no adequate initial guess is supplied. In contrast, an equivalent

interval-based algorithm is guaranteed to find the global optimum without requiring an initial guess and without linearizing the nonlinear function.

1.1.2 Assumptions

In the following, we discuss the assumptions and framework conditions of this work as well as the challenges that arise from it.

Off-the-shelf sensors

This work introduces sensor error models for cameras, laser scanners and IMUs. As we assume these sensors to be available off the shelf, several restrictions must be taken into account. First, commercially available sensors are typically mass produced, limiting the possibility of thorough calibration of each sensor for cost reasons. Custom-made sensors, on the other hand, are generally much more expensive, but subject to extensive calibration. Thus, their error distribution is usually known, whereas this is not the case with off-the-shelf sensors.

Second, commercially available sensors are typically black-box systems that do not allow to access the raw data. In addition, they usually do not provide interfaces for the time synchronization or the extrinsic calibration with other sensors. Consequently, the spatiotemporal calibration parameters can only be determined using the measurement data that is provided by the sensor. In addition, the measurement data itself may already be influenced by a variety of different types of errors when transmitted to the user. For example, a laser scanner typically provides a distance, but determines that distance by measuring the Time of Flight (ToF) of a laser beam. Therefore, it should be noted that when developing error models for these black-box sensors, an error can only be assigned to the data provided by the sensor, but this error consists of several different error sources.

Sensor errors

We assume to know bounds enclosing the maximum error of each sensor. These bounds can be established in experiments, specified by the manufacturer or derived from physical fundamentals. The only requirement is that the actual error does not exceed these bounds. However, sporadic outliers, i.e. errors exceeding the bounds, are allowed and can also be taken into account using interval analysis. These outliers can occur because of faults in the measurement process, environmental influences, or due to software errors (e.g. mismatched image features). Nonetheless, to take these outliers into account and preserve the guarantees of interval analysis, we require knowledge about the maximum number of outliers (e.g. no more than $5\,\%$ of image features are mismatched). Naturally, the number of outliers must not exceed $50\,\%$. Often, the maximum number of outliers is difficult to estimate. Therefore, in case of doubt, this number as well as the error bounds should be chosen rather conservatively (i.e. more tolerant) to avoid sensor errors violating these bounds.

Clock synchronization

Although a contribution of this work is an approach to fuse data from multiple sensors under consideration of spatiotemporal uncertainties, we assume the data streams of camera and laser scanner to be synchronized in time and only model the extrinsic calibration between these sensors as uncertain. The reasoning behind this is that a potential timestamp offset raises the question how to fuse the information from camera and laser scanner in a guaranteed way if the environment is not captured at the same time by both sensors. This question is difficult since a simple linear temporal interpolation of, for example, the laser scan points does not suffice due to the environment being discontinuous (i.e. the distance of a single laser scan point does not in-/decrease linearly).

Moreover, for simplicity we assume only a constant timestamp offset between camera and IMU. Theoretically, however, this timestamp can be repeatedly re-estimated so that the effect of drifting sensor clocks is negated.

Sensor positioning

For sensor positioning, we assume that the camera and laser scanner are close, i.e. their baseline is small. If this were not the case, their view of objects in the environment would be significantly different, resulting in cases where the camera is able to detect an object, but the laser scanner is not, or vice versa. Consequently, we would not be able to fuse information for this specific object. Naturally, this also requires the objects in the environment to be detectable by both sensors, thereby excluding materials such as glass that might be visible in a camera image due to reflections, but are invisible to a laser scanner due to rays passing through the glass. Moreover, we assume all sensors to be mounted rigidly so that the IMU is able to measure the same rotations experienced by the other sensors.

1.2 Solution approach

This work introduces a new approach to fuse information from camera, laser scanner and IMU for the dead reckoning of a mobile robot. However, before we can fuse information from the different sensors we have to understand their error characteristics and model them accordingly. Within this work, we identify interval analysis as an appropriate tool to model the errors of all three sensors since it allows to propagate sensor errors consistently and in a guaranteed way. Together with the other advantages of interval analysis mentioned above, this is important for safety-critical systems. Consequently, we propose bounded-error models for all three sensors that take different types of error into account.

The newly introduced bounded-error models are the first step to guarantee the integrity of fused sensor information. Next, we also require the inter-sensor properties such as a timestamp offset or the extrinsic transformation between sensor coordinate systems before information from different sensors can be fused. However, these spatiotemporal calibration parameters cannot be determined exactly, thus introducing an additional source of error for the sensor fusion that needs to be modeled accordingly. In contrast to the sensor errors, the error of these parameters is purely systematic since we assume the timestamp offset to be constant and the

sensors to be rigidly mounted. Consequently, an error in the estimation of the spatiotemporal calibration parameters is also constant and therefore systematic. As mentioned above, unlike stochastic error models, the interval-based error model is consistent with systematic errors. Thus, we use intervals to enclose the true spatiotemporal calibration parameters for the sensor fusion.

However, it remains to determine these intervals. One possibility is to use external, more accurate devices (e.g. a laser tracker) to measure the transformation between sensor coordinate systems and then estimate conservative interval bounds that are guaranteed to enclose the true parameters. Similarly, the timestamp offset can be determined using conventional algorithms such as the Network Time Protocol (NTP) [15]. However, this requires additional knowledge and/or cooperation from the sensors and cannot be done for black-box systems. Therefore, we must use the measurement data provided by the sensor to determine these parameters. However, the existing approaches using measurement data rely on a stochastic error model, thus possibly underestimating the calibration error due to the reasons mentioned above.

To determine the spatiotemporal calibration parameters in a guaranteed way without relying on external information or stochastic estimates, we propose interval-based approaches. First, we compute the timestamp offset and the extrinsic rotation between the camera and the IMU. For this approach, we rotate the sensor setup in front of a calibration target, determine the intervals enclosing each individual sensor's rotation over time, and finally determine the calibration parameters by aligning the rotation tubes (intervals over time) using a newly developed so-called contractor $\mathcal{C}_{\text{offset}}$. Second, we find the 6 Degrees Of Freedom (DOF) extrinsic transformation between the camera and laser scanner coordinate systems. For this approach, we position a calibration target in different poses in front of the sensor setup. Subsequently, we introduce novel algorithms to detect the calibration target in both both camera and laser scanner data which introduces constraints for the desired calibration parameters. Afterwards, based on these constraints we build contractors that are finally employed in conjunction with the SIVIA algorithm to compute intervals for the extrinsic calibration parameters.

After the spatiotemporal calibration parameters and the sensor errors are modeled properly, we focus on the aforementioned sensor fusion. First, we propose a new algorithm to fuse information from camera and laser scanner by finding distinct image features and assigning distance information from the laser scanner to them. Here, a main contribution of this work is the consideration of both sensor and inter-sensor errors. Next, these fused 3D points are employed to determine the robot's rigid body transformation between consecutive points in time, corresponding to the desired dead reckoning. Simultaneously, angular velocities measured by the IMU are used to further constrain the robot's rotation.

The computed interval poses are guaranteed to contain the robot's true pose since all possible sensor and calibration errors are modeled accordingly. Naturally, the underlying assumption is that the assumed error bounds are correct. If this is the case, interval analysis allows us to propagate the errors consistently and without dismissing a potential solution. Our approach is therefore suitable for safety-critical systems, such as autonomous cars, for which the maximum error is more important than the average error.

Finally, we evaluate all approaches experimentally and compare their results to stochastic methods. Simulated data allows us to access ground truth information so that we can verify

whether our approaches enclose the true result. Moreover, it enables us to examine different error distributions and unknown systematic errors with regard to their influence on our interval-based and a conventional stochastic error model. Contrary, real data allows us to evaluate whether our approaches are suitable for use with real sensors and real robots. Furthermore, a comparison with existing stochastic approaches demonstrates possibilities for the joint use of both the interval-based and stochastic error model for safety-critical systems.

1.3 Document structure

In the following we detail the structure of this work. Fig. 1.1 provides a graphical overview in which the links between the contributions of this work are clarified. We refrain from explaining this figure for the time being, as we will show it several times throughout the document so that the reader can follow the common thread of this work and finally understand the contents of the figure.

In **Chapter 2** we summarize the current state of research and introduce basics required for this work.

Chapter 3 first introduces the basics of interval analysis and subsequently depicts other interval-based work that is related to ours.

In **Chapter 4** we detail various types of error for each sensor and investigate these error sources with regard to their compatibility with a stochastic or a bounded error model. Based on our findings, we introduce error models for all three sensors that assume unknown but bounded errors.

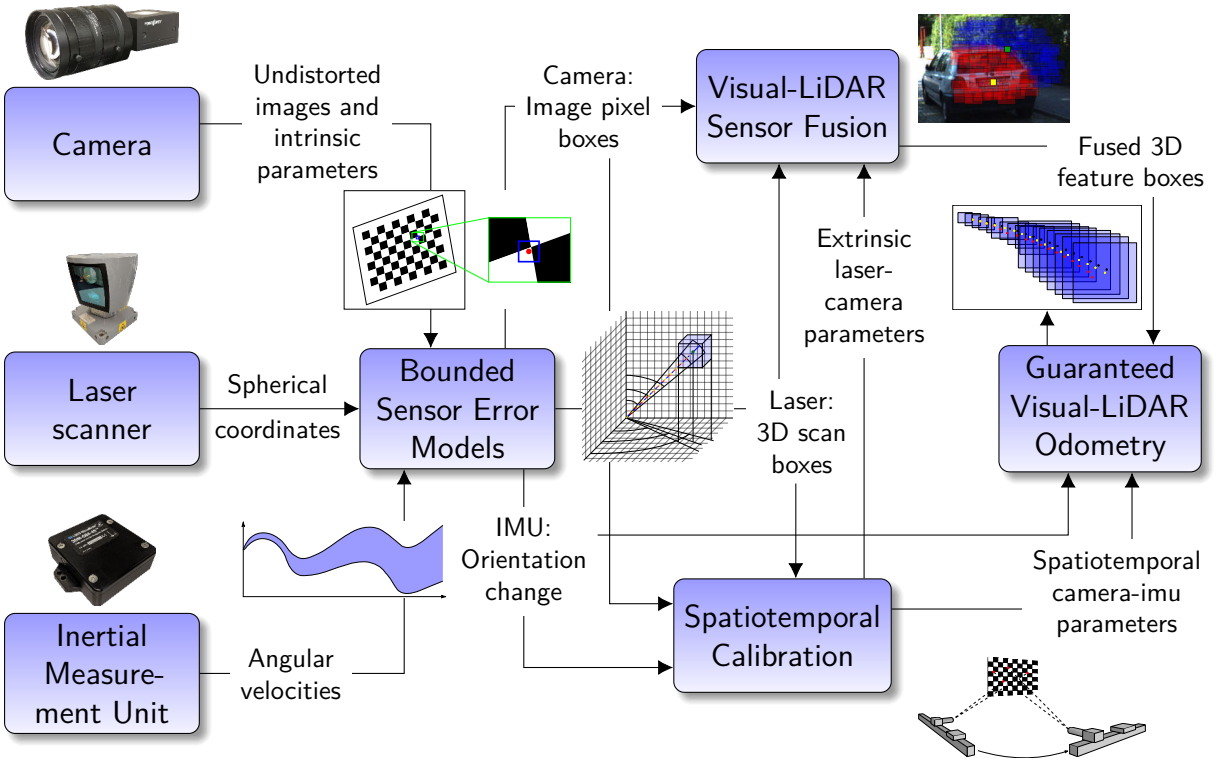


Figure 1.1: Overview of the structure of this work.

Chapter 5 introduces our interval-based approach to solve the PnP problem. This constitutes preliminary work that is necessary for the following methods proposed in **Chapter 6**. Here, we detail two approaches for the spatiotemporal calibration of the sensors. First, we focus on the spatiotemporal calibration between camera and IMU, providing a method to compute both the time offset between sensor clocks and the extrinsic rotation between sensor coordinate systems. Second, we introduce an approach to perform the six DOF extrinsic calibration between camera and LiDAR, thus completing the spatiotemporal calibration of our sensor setup.

In **Chapter 7** we introduce an approach for the visual-LiDAR odometry of a mobile robot that fuses information from camera, laser scanner and IMU under consideration of the previously determined spatiotemporal calibration parameters. This chapter is divided into two parts. First, we explain how to fuse information from camera and laser scanner in a guaranteed way. Second, we detail how to employ this fused information alongside angular velocities measured by the IMU to compute the robot's pose in 3D.

Chapter 8 provides an extensive experimental evaluation of all introduced approaches using simulated and real data. Here, a particular emphasis is put onto the comparison with conventional stochastic methods.

In **Chapter 9** we discuss and assess the results obtained in this work. In addition, we give an outlook on open research questions and future work.

Finally, **Chapter 10** concludes this work with a summary.

2 State of the Art

In the following, we first introduce basic requirements for this work. Next, we present state-of-the-art approaches for the spatiotemporal calibration of multi-sensor systems and the dead reckoning of mobile robots with a special focus on methods that fuse information from camera, laser scanner and IMU. In this chapter, however, we focus only on approaches using stochastic error modeling, whereas all related work that uses interval-based error modeling is depicted in Chapter 3.

2.1 Sensor models

In the following, we introduce the sensors employed in this work. This introduction includes an explanation of the operating principle and a model of each sensor. The sensors we employ are a 3D laser scanner, a color camera, and an IMU.

2.1.1 3D laser scanner

This work employs 3D laser scanners for the perception of the environment. Laser scanners can be classified as active, optical measurement systems since they emit and subsequently detect light. In the simple case of measuring the distance to a single point, a laser ray (typically at a wavelength of 905 nm) is emitted by the sensor and diffusely reflected by the target in the environment. Part of the light is reflected to the sensor, and can thus be detected by a photodiode (receiver). Now, the distance to the measured point can be computed by considering the speed of light and the time that passed between the emission and the detection of the laser ray. The corresponding time period is also known as the ToF. Due to this measurement technique, laser scanners are also denoted as LiDAR.

To extend the idea of the one-dimensional distance measurement to the perception of a three-dimensional environment, multiple approaches have been considered. In this work, we use laser scanners manufactured by Velodyne [1]. They consist of multiple (e.g. 16 or 64) emitters and receivers that are arranged on top of each other such that they scan along a virtual vertical line. Thus, the vertical field of view is limited by the number of emitters and their respective angular distance. Furthermore, this whole optical system is rotated at a high speed to provide a 360° horizontal field of view.

[1] https://velodynelidar.com/

Still, the only measurement provided by each emitter-receiver-pair is a simple distance r. To compute a 3D point from this measurement, it is inevitable to know the polar (vertical) angle θ and the azimuthal (horizontal) angle φ. As the single laser sensors are arranged on top of each other, the vertical angle is constant throughout all measurements and can be found in the data sheet. To obtain the horizontal angle, the laser scanner measures its current rotation whenever performing a laser measurement. Thus, finally, we can compute the 3D coordinates of any measured point by converting the spherical coordinates r, θ and φ to Cartesian coordinates:

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} r \cdot \sin\theta \cdot \cos\varphi \\ r \cdot \sin\theta \cdot \sin\varphi \\ r \cdot \cos\theta \end{pmatrix}. \tag{2.1}$$

However, as the manufacturing process of the laser scanner is not perfect and the positioning of the emitter-receiver-pairs might not be accurate, deviations can occur for all three spherical coordinates. To account for these deviations, a calibration of the laser scanner must be performed during which a multitude of correction parameters are determined [16].

For this work, we assume that this calibration is carried out beforehand and the correction parameters are determined appropriately. However, these correction parameters cannot be computed perfectly, and thus deviations can still occur. Among other things, they therefore constitute a source of error, which is modeled accordingly in Chapter 4.

2.1.2 Camera

In this work, we employ a color camera that is modeled using the *pinhole camera model* [17], which is the simplest yet most common camera model. The pinhole camera model describes the mapping between the environment and the image taken in the camera coordinate system C. For this purpose, the camera aperture, through which the light emitted by the environment passes on the way to the image sensor, is idealized as a point. This point is coincident with the origin of the camera coordinate system C. Fig. 2.1 shows the pinhole camera model. To simplify the visualization, the image plane (depicted in orange) is projected in front of the pinhole, although in reality it resides behind the pinhole resulting in an image that is turned by 180°.

As can be seen, the camera coordinate system is oriented such that the Z_C-axis corresponds to the optical axis. Furthermore, the plane spanned by X_C and Y_C is parallel to the image plane spanned by u and v with the principal point (c_x, c_y). Besides, the image plane (orange) is discretized into small pixels whose number yields the image resolution. Now, the light ray emitted by the point $\mathbf{X}$ passes through the pinhole at the camera's optical center. On its way it intersects the image plane where it causes the image coordinates (u, v) to be colored respectively. The process of determining the color of a pixel is known as the imaging process and is performed by the image sensor. There exist multiple types of image sensors, the most popular being Charge-Coupled Device (CCD) and Complementary Metal-Oxide Semiconductor (CMOS) sensors. For example, the operation of a CCD sensor to discretize the continuous environment can be compared to placing an array of buckets (the discretized pixels) on a field (the continuous environment) to measure the spatial distribution of rainfall [18].

To further visualize the following mathematical derivations, Fig. 2.2 shows the pinhole camera model as seen from the X_C-axis. At first, we introduce the projection of 3D points into the two-dimensional image plane:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.2}$$

where (X, Y, Z) are the coordinates of a 3D point in the camera coordinate system, (u, v) are the image coordinates in pixel, f_x and f_y are the focal lengths in pixel units, (c_x, c_y) is the principal point that is approximately at the image center, and $\mathbf{K}$ is the camera matrix consisting of the intrinsic parameters. Usually, these intrinsic parameters are determined during camera calibration by employing a calibration target (e.g. a checkerboard) of known dimensions [17]. Furthermore, λ is the unknown scale factor since the camera is only able to measure the direction of a point, but not its absolute distance. Thus, we introduce the notion of *normalized (or homogeneous) image coordinates* which describe the direction of a 3D point by dividing its coordinates by Z:

$$\tilde{\mathbf{X}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{pmatrix}, \quad \text{with} \quad \tilde{x} = \frac{X}{Z}, \quad \tilde{y} = \frac{Y}{Z}. \tag{2.3}$$

Consequently, this allows us to directly express the image coordinates (u, v):

$$\begin{aligned} u &= f_x \, \tilde{x} + c_x, \\ v &= f_y \, \tilde{y} + c_y. \end{aligned} \tag{2.4}$$

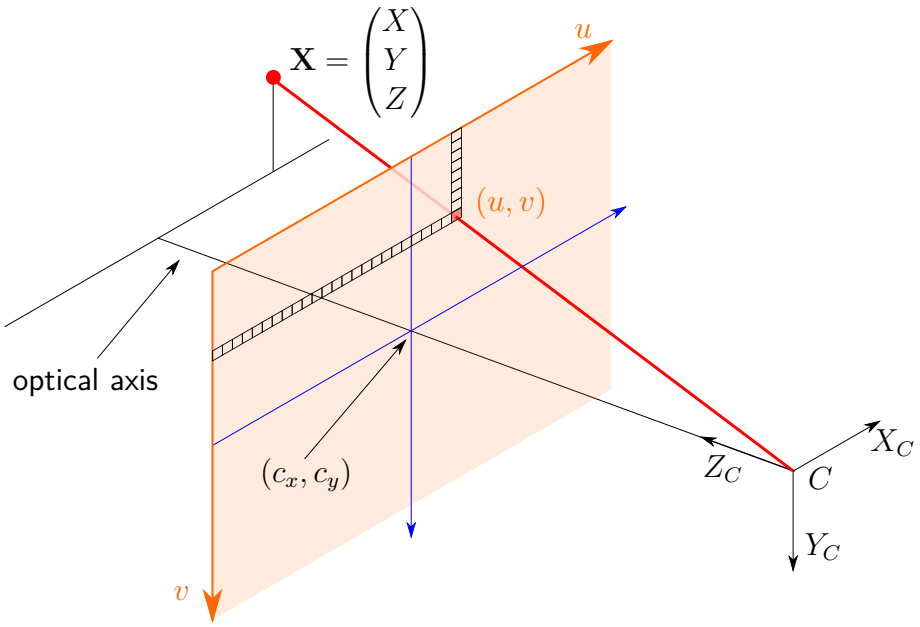


Figure 2.1: Illustration of the pinhole camera model.

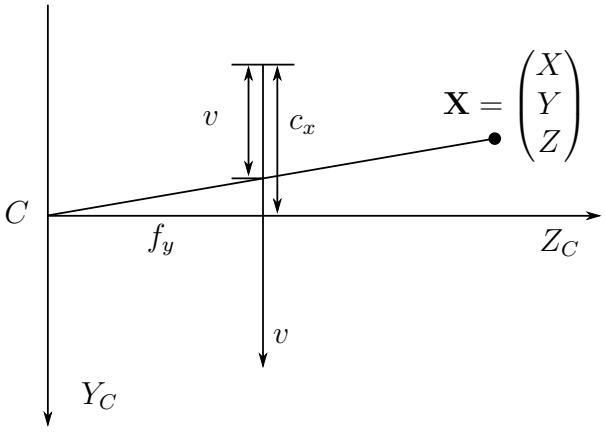
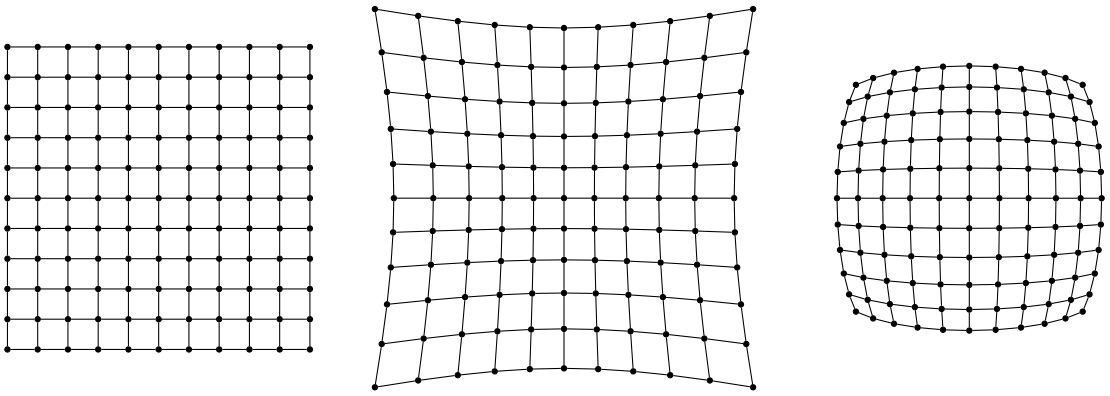


Figure 2.2: The pinhole camera model as seen from the X_C-axis.



(a) No radial distortion. (b) Pincushion distortion. (c) Barrel distortion.

Figure 2.3: Classification of different radial distortions.

Likewise, we can also compute the normalized image coordinates given the image coordinates:

$$\begin{aligned} \tilde{x} &= \frac{u - c_x}{f_x}, \\ \tilde{y} &= \frac{v - c_y}{f_y} \end{aligned} \tag{2.5}$$

Generally, however, lenses are attached to the camera, resulting in distortions that are not modeled by the pinhole camera model. Fig. 2.3 shows the effect of radial distortions on a grid pattern. Nevertheless, this distortion can be removed once the distortion parameters k_1, k_2, k_3, p_1 and p_2 have been computed during camera calibration:

$$\begin{aligned} r^2 &= \tilde{x}^2 + \tilde{y}^2, \\ \tilde{x'} &= \tilde{x}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 \tilde{x}\tilde{y} + p_2\left(r^2 + 2\tilde{x}^2\right), \\ \tilde{y'} &= \tilde{y}\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1\left(r^2 + 2\tilde{y}^2\right) + 2p_2 \tilde{x}\tilde{y}, \end{aligned} \tag{2.6}$$

where k_1, k_2, k_3 are radial distortion parameters and p_1, p_2 are tangential distortion parameters.

After removing the distortion, the new image coordinates (u', v') can be computed accordingly:

$$\begin{aligned} u' &= f_x \, \tilde{x'} + c_x, \\ v' &= f_y \, \tilde{y'} + c_y. \end{aligned} \tag{2.7}$$

For this work, we assume to be able to compute the distortion parameters reasonably accurate, such that the distortion can be removed from all images before further processing them. This assumption holds since the distortion depends on the lens, and is thus - in contrast to the intrinsic parameters which can be adapted by the user (e.g. the focal length) - constant over time as modern lenses are manufactured rigidly. Furthermore, we employ camera lenses with low distortions (i.e. no fisheye lenses).

2.1.3 Inertial Measurement Unit (IMU)

An IMU is an assembly of multiple inertial sensors. Usually, it contains three orthogonally mounted accelerometers and gyroscopes (six sensors total) to measure accelerations and angular velocities around all three coordinate system axes. Optionally, it can also include a magnetometer to measure the direction relative to the geographic cardinal directions. In this work, we only employ data from the gyroscopes, and thus provide an introduction for this sensor only.

A gyroscope measures the rotational velocity around one axis at a high rate (e.g. 100 Hz). By mounting three of them orthogonally, the IMU is able to provide the angular velocity around all three axes resulting in a 3×1 vector $\tilde{\boldsymbol{\omega}} = (\, \tilde{\omega}_x \;\; \tilde{\omega}_y \;\; \tilde{\omega}_z \,)^\intercal$. However, the measurements are distorted by many different error sources that can be modeled and propagated according to [19]. Using a simple error model the true angular velocity vector can be expressed as

$$\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} + \Delta_s \cdot \tilde{\boldsymbol{\omega}} + \boldsymbol{\Delta}_b + \boldsymbol{\Delta}_n, \tag{2.8}$$

where Δ_s is the scale factor (linear relation between true value and measurement), $\boldsymbol{\Delta}_b$ is the constant bias and $\boldsymbol{\Delta}_n$ is the measurement noise. Traditionally, these errors are modeled using stochastic processes and propagated accordingly [19]. We further detail these types of error and develop our own interval-based error model for them in Chapter 4.

At the beginning of each experiment, the IMU has to remain static for a few seconds to define the so called *initial alignment* (i.e. the pose relative to which we compute the orientation in the following) and estimate the biases [20]. To compute the orientation relative to this initial alignment, the rotational velocity vector $\boldsymbol{\omega}$ must be integrated. However, as the angular rates are measured in the continuously changing body frame of the sensor, a simple direct integration is not possible. Instead, we have to integrate the ordinary differential equation

$$\dot{\mathbf{R}}_{I_t}^{I_0} = \mathbf{R}_{I_t}^{I_0} \cdot \left(\langle \boldsymbol{\omega}_t \rangle_\times \right), \tag{2.9}$$

where $\boldsymbol{\omega}_t$ is the angular velocity measurement at time t and $\langle \boldsymbol{\omega}_t \rangle_\times$ is the skew-symmetric matrix of $\boldsymbol{\omega}_t$.

This skew-symmetric matrix of any vector $\boldsymbol{\omega} = (\,\omega_x \;\; \omega_y \;\; \omega_z\,)^\intercal$ is defined as

$$\langle \boldsymbol{\omega} \rangle_\times = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix}. \tag{2.10}$$

Since the integration of this ordinary differential equation is difficult, it is often simplified and computed stepwise. Consequently, we first compute the small, incremental rotation $\mathbf{R}^{I_t}_{I_{t+\Delta t}}$ between two subsequent time instants t and $t + \Delta t$ and subsequently apply the rotation that occurred up until t:

$$\mathbf{R}^{I_0}_{I_{t+\Delta t}} = \mathbf{R}^{I_0}_{I_t} \cdot \mathbf{R}^{I_t}_{I_{t+\Delta t}}. \tag{2.11}$$

To determine the small, incremental rotation $\mathbf{R}^{I_t}_{I_{t+\Delta t}}$, we follow the pre-integration approach described in [21]. Here, the authors assume that the angular velocity is constant for Δt. This assumption can be made since the IMU measures at a high rate, and thus Δt is small (e.g. $\Delta t = 0.01\,\mathrm{s}$). From the kinematic model [20] it follows that

$$\mathbf{R}^{I_0}_{I_{t+\Delta t}} = \mathbf{R}^{I_0}_{I_t} \cdot \exp\left(\langle \boldsymbol{\omega}_t \rangle_\times \, \Delta t\right), \tag{2.12}$$

where $\boldsymbol{\omega}_t$ is the angular velocity measurement at time t and $\langle \boldsymbol{\omega}_t \rangle_\times$ is the skew-symmetric matrix of $\boldsymbol{\omega}_t$.

To compute $\exp\left(\langle \boldsymbol{\omega} \rangle_\times\right)$ for any vector $\boldsymbol{\omega} = (\,\omega_x \;\; \omega_y \;\; \omega_z\,)^\intercal$, let $\theta = \sqrt{w_x^2 + w_y^2 + w_z^2}$. According to the Rodrigues formula it follows that

$$\exp\left(\langle \boldsymbol{\omega} \rangle_\times\right) = \mathbf{I}_3 + \frac{\sin\theta}{\theta} \langle \boldsymbol{\omega} \rangle_\times + \frac{(1 - \cos\theta)}{\theta^2} \langle \boldsymbol{\omega} \rangle_\times^2. \tag{2.13}$$

While this first-order integration works well for high-rate gyroscope measurements, higher order integration methods should be employed if the measurement frequency is low [22].

2.2 Stochastic error modeling

In reality, no sensor is capable of performing error-free measurements due to various factors such as physical effects, changing environments, or manufacturing uncertainties of the sensor. This results in an error e that distorts the measurement of x which is the true value we are interested in such that

$$y = f(x) + e, \tag{2.14}$$

where f is the known measurement function and y is the distorted value our sensor measures. Unfortunately, e is generally unknown, and thus we cannot compute the true value x given the measurement y.

Nevertheless, sensor data has to be employed as there is virtually no other possibility to gather knowledge about, for example, a robot's pose. In order to do that, the unknown error e must be computed as precisely as possible and the remaining uncertainty must be assessed. In the ideal case, this allows to compute an estimation of the true value x while simultaneously also providing an assessment of the accuracy of this estimation. Therefore, the error must

be modeled mathematically while taking all possible types of error into account. In practice, however, this is not possible since there are many different types of error that may be too complex to model or that require knowledge of unknown characteristics of the environment (e.g. the surface material) [23]. Thus, generally the goal is to establish an error model that takes the most important sources of error into account and approximates reality as accurately as possible. It follows that the more accurate the sensor model, the better the results of the measurement [23].

Especially since the publication of the book "Probabilistic Robotics" by Sebastian Thrun et al. [23], the most common error model for sensor inaccuracies in robotics is based on probability theory. Here, the basic assumption is that sensor errors can be described using a probability distribution for which the mean value corresponds to the true value, i.e. when measuring, for example, a distance infinitely often, the mean over all measurements is the true distance. In contrast, a single measurement can deviate from this mean value and the probability of such a deviation is indicated by the probability distribution.

The most commonly employed probability distribution is the continuous normal distribution (or Gaussian distribution):

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \tag{2.15}$$

where μ is the mean value and σ is the standard deviation. Assuming this distribution for the error introduced in (2.14) allows us to state that the error e is distributed normally with zero mean and a variance of σ^2, i.e. $e \sim \mathcal{N}(0,\, \sigma^2)$.

As previously explained, however, a measurement is generally not affected by only one type of error, but suffers from different error sources that must be modeled accordingly. In this case, each error is modeled using a distinct probability distribution. Consequently, probability theory allows to straightforwardly combine these distributions to one common error distribution under the assumption of independent errors. For example, Thrun et al. [23] introduce a beam model of range finders that takes four error sources (measurement noise, unexpected objects, failures, random measurements) into account and combines the four different distributions by mixing them by a weighted average.

After finding an appropriate probability distribution for the error of each sensor, stochastic methods are employed to propagate these errors from the input sources (sensors) to the output sources (e.g. localization result) [24].

2.3 Rotation parametrization

Since this work deals with sensor calibration and localization in 3D, rotations must also be modeled in 3D. The most basic, unambiguous representation for a three-dimensional rotation is the *rotation matrix*. A 3×3 matrix $\mathbf{R}$ is a properly termed rotation matrix if and only if $\mathbf{R}^\intercal = \mathbf{R}^{-1}$ and $\det \mathbf{R} = 1$.

Using rotation matrices, subsequent rotations can be computed straightforwardly by multiplying the corresponding rotation matrices. For example, $\mathbf{R} = \mathbf{R}_2 \cdot \mathbf{R}_1$ is the rotation given by $\mathbf{R}_1$ followed by the rotation given by $\mathbf{R}_2$.

However, rotation matrices pose two major disadvantages. First, they cannot be understood intuitively. Second, they possess nine entries although three variables are sufficient to express a three-dimensional rotation since $\mathbf{R} \in SO(3)$, where $SO(3)$ is the 3D rotation group. Thus, there exist several other representations of rotations, of which we employ only two, and consequently we introduce only these two in the following.

2.3.1 Euler angles

The coordinate frame of a rigid body in 3D are defined by the three coordinate axes x, y and z. On this basis, in 1770 Euler proposed to describe the rotation of a rigid body by defining the rotation around the three coordinate axes individually. However, this formalism creates ambiguities as the order in which the rotations are performed around the axes is important. In this work, we adopt the roll-pitch-yaw formulation as it is the accepted standard in mobile robotics. Here, roll is the rotation around the x-axis, pitch is the rotation around the y-axis and yaw is the rotation around the z-axis. In this work, we denote the vector of Euler angles as $\boldsymbol{\xi} = (\,\varphi \;\; \theta \;\; \psi\,)^\intercal$, where φ is the yaw angle, θ is the pitch angle and ψ is the roll angle.

Given the three Euler angles $\boldsymbol{\xi}$, the corresponding rotation matrix $\mathbf{R}$ can be computed as

$$\begin{aligned}\mathbf{R}(\boldsymbol{\xi}) &= \mathbf{R}(\psi) \cdot \mathbf{R}(\theta) \cdot \mathbf{R}(\varphi) \\ &= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta\cos\psi & -\cos\varphi\sin\psi + \sin\theta\cos\psi\sin\varphi & \sin\psi\sin\varphi + \sin\theta\cos\psi\cos\varphi \\ \cos\theta\sin\psi & \cos\psi\cos\varphi + \sin\theta\sin\psi\sin\varphi & -\cos\psi\sin\varphi + \sin\theta\cos\varphi\sin\psi \\ -\sin\theta & \cos\theta\sin\varphi & \cos\theta\cos\varphi \end{pmatrix}.\end{aligned} \tag{2.16}$$

Conversely, given a rotation matrix $\mathbf{R}$, we can compute the three Euler angles as

$$\begin{aligned}\varphi &= \arctan2(R_{32}, R_{33}), \\ \theta &= -\arcsin R_{31}, \\ \psi &= \arctan2(R_{21}, R_{11}),\end{aligned} \tag{2.17}$$

where R_{ij} is the entry in the i-th row and j-th column of $\mathbf{R}$.

However, the main drawback of the Euler angles is that a so called *gimbal lock* can occur. For the formulation we chose, the gimbal lock occurs when $\theta = \frac{\pi}{2}$ (or generally when $\cos\theta = 0$). This means that two of the three axes become collinear (i.e. after applying the rotation of $\theta = \frac{\pi}{2}$, the new x-axis coincides with the previous z-axis). In this case, one degree of freedom is lost, and thus not all possible rotations can be represented.

Furthermore, two rotations expressed in Euler angles cannot be combined straightforwardly, but must be converted into rotation matrices first.

Another issue is that the Euler angles are periodic since sine and cosine are periodic. Consequently, their range must be limited such that, for example, $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\varphi \in [-\pi, \pi]$ and $\psi \in [-\pi, \pi]$. However, this becomes problematic when dealing with bounded uncertainties

for the Euler angles. For example, we might want to express that $\varphi \in ([-\pi, -3] \cup [3, \pi])$ which will be equal to $\varphi \in [-\pi, \pi]$ when dealing with intervals.

2.3.2 Modified Rodrigues Parameters (MRP)

In contrast to the Euler angles, the MRP have been introduced comparatively recent [25]. Similar to the Euler angles, they are triplets in $\mathbb{R}^3$ and are denoted as a 3×1 vector $\boldsymbol{\rho}$ in this work. However, they are more closely related to the axis-angle representation, which usually represents a rotation by a rotation axis and the magnitude of rotation around this axis. For example, a well known axis-angle representation are the quaternions [26].

In contrast to other rotation representations, the MRP have the advantage that they consist of only three scalars, that their singular points are as far away from the origin as possible and that two subsequent rotations can straightforwardly be combined.

In the following, we give a mathematical introduction of the MRP, that we previously published in [27].

Let $\mathbf{q}$ be a quaternion [26], that is defined as

$$\mathbf{q} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \mathbf{q_{13}} \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ \boldsymbol{\eta} \sin(\theta/2) \end{pmatrix}, \tag{2.18}$$

where $\boldsymbol{\eta}$ is a unit vector representing the rotation axis and θ is the angle of rotation. The 3×1 MRP vector is defined as

$$\boldsymbol{\rho} = \frac{\mathbf{q_{13}}}{1 + q_0} = \boldsymbol{\eta} \tan(\theta/4). \tag{2.19}$$

The 3×3 rotation matrix for $\boldsymbol{\rho}$ is

$$\mathbf{R}(\boldsymbol{\rho}) = \mathbf{I}_3 + \frac{8 \langle \boldsymbol{\rho} \rangle_\times^2 + 4 \left(1 - \|\boldsymbol{\rho}\|^2\right) \langle \boldsymbol{\rho} \rangle_\times}{\left(1 + \|\boldsymbol{\rho}\|^2\right)^2}, \tag{2.20}$$

where $\langle \boldsymbol{\rho} \rangle_\times$ is the skew-symmetric matrix of $\boldsymbol{\rho}$ and $\|\boldsymbol{\rho}\|$ is the norm of $\boldsymbol{\rho}$.
Sequential rotation by $\boldsymbol{\phi}$ and then by $\boldsymbol{\rho}$ is denoted by the bullet operator ($\bullet$) and defined as

$$\boldsymbol{\rho} \bullet \boldsymbol{\phi} = \frac{\left(1 - \|\boldsymbol{\rho}\|^2\right) \boldsymbol{\phi} + \left(1 - \|\boldsymbol{\phi}\|^2\right) \boldsymbol{\rho} - 2 \langle \boldsymbol{\phi} \rangle_\times \boldsymbol{\rho}}{1 + \|\boldsymbol{\rho}\|^2 \|\boldsymbol{\phi}\|^2 - 2 \boldsymbol{\rho}^\intercal \boldsymbol{\phi}}. \tag{2.21}$$

Another useful property of the MRP is that

$$\mathbf{R}^\intercal(\boldsymbol{\rho}) = \mathbf{R}(-\boldsymbol{\rho}), \tag{2.22}$$

which allows direct computation of the inverse rotation.

Finally, using the MRP it is possible to directly express the kinematic differential equation that is required to integrate, for example, angular rates that are measured by an IMU (cf. Section 2.1.3):

$$\dot{\boldsymbol{\rho}}(t) = \frac{1}{4}\left(\left(1 - \|\boldsymbol{\rho}(t)\|^2\right)\mathbf{I}_3 + 2\,\langle\boldsymbol{\rho}(t)\rangle_\times + 2\,\boldsymbol{\rho}(t)\boldsymbol{\rho}^\intercal(t)\right)\boldsymbol{\omega}(t), \tag{2.23}$$

where $\boldsymbol{\omega}(t)$ are the angular velocities at time t and $\boldsymbol{\rho}(t)$ are the MRP describing the rotation relative to the initial alignment.

2.4 Perspective-n-Point (PnP) problem

The PnP problem concerns the estimation of the pose of a camera in a local world coordinate system given distinct 2D features in the image and their corresponding 3D coordinates in the world coordinate system. It has to be solved mainly for the intrinsic and extrinsic calibration of cameras, but is also relevant in the field of 3D pose estimation relative to known landmarks. Consequently, the PnP problem is studied in both the Computer Vision [28] and the Photogrammetry [29] communities.

The PnP problem can be formalized as follows. Given a set of n image coordinates (u_i, v_i) in pixels and the corresponding 3D world coordinates $\mathbf{X}_i^W = (\,X_i^W \;\; Y_i^W \;\; Z_i^W\,)^\intercal$ with $i \in \{1, \ldots, n\}$, the relation between the camera coordinate system C and the world coordinate system W can be formulated as:

$$\lambda \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \mathbf{K}\left(\mathbf{R}_W^C \mathbf{X}_i^W + \mathbf{T}_W^C\right), \tag{2.24}$$

where λ is the unknown scale factor, $\mathbf{K}$ is the camera intrinsic matrix and $\mathbf{R}_W^C$ and $\mathbf{T}_W^C$ are the desired extrinsic rotation matrix and translation vector, respectively, describing the pose of the camera in the world coordinate system W. Generally, the camera intrinsic matrix $\mathbf{K}$ is assumed to be known.

To solve the PnP problem, at least three corresponding points are required, i.e. $n = 3$. This minimal form of the PnP problem is also denoted as P3P and requires to solve a three-dimensional non-linear equation system [30]. Due to the nature of this system, up to four solutions are possible. However, not all four solutions are geometrically feasible (e.g. if the solution implies that the world points are not within the field of view of the camera). Generally, such additional constraints lead to a single solution for the extrinsic parameters. If this is not the case, additional knowledge must be acquired (e.g. a fourth point).

If at least four corresponding point pairs are available, the EPnP approach introduced by Lepetit et al. [31] can be employed. It is the first algorithm that provides an accurate solution in $O(n)$. In order to do that, the n 3D points are expressed as a weighted sum of four virtual control points. Accordingly, the unknowns to solve for are these virtual control points. To determine their coordinates, a small constant number of quadratic equations has to be solved.

An extension of the EPnP approach is the UPnP approach published by Kneip et al. [32]. It does not require the camera intrinsic parameters to be known beforehand and extends the idea of the EPnP algorithm to non-central cameras. Furthermore, this approach prevails in the case of solution multiplicity (i.e. if multiple poses are possible solutions).

If even more redundant points are available ($n \geq 6$), the Direct Linear Transformation (DLT) [33] algorithm can be employed to solve for twelve unknowns (nine entries of the rotation matrix and three entries of the translation vector) while ignoring the properties of the rotation matrix. Although this approach cannot obtain an exact solution, it is often used to determine an initial guess with little computational effort.

Besides the presented non-iterative solutions to the PnP problem, iterative optimization techniques like the Levenberg-Marquardt algorithm [14] can be employed to minimize the reprojection error. While iterative algorithms generally achieve a high accuracy, they are computationally expensive and instable due to the local minima of the cost function [34]. Thus, non-iterative approaches can be employed to find an initial solution that is further optimized using an iterative method.

2.5 Spatiotemporal calibration

When employing multi-sensor systems to combine the advantages of different sensor by fusing information from them, it is important to accurately know the spatiotemporal parameters relating the information from the sensors. As indicated by the term "spatiotemporal", there exist spatial parameters such as the extrinsic transformation between sensor coordinate systems and temporal parameters such as the time offset and the drift between sensor clocks. Finding these spatial and temporal parameters is denoted as the *spatiotemporal calibration* of the multi-sensor system.

While it is convenient to estimate the spatial and temporal parameters simultaneously, it is also possible to perform a spatial (i.e. extrinsic) and temporal calibration separately. The extrinsic calibration concerns the extrinsic transformation between sensor coordinate systems which generally consists of a rotation and a translation. Here, the goal is to determine a rotation matrix $\mathbf{R}_B^A \in \mathbb{R}^{3\times3}$ and a translation vector $\mathbf{T}_B^A \in \mathbb{R}^{3\times1}$ that allows to transform a 3D point $\mathbf{X}^B$ measured in the sensor coordinate system B into the coordinate system of the other sensor A:

$$\mathbf{X}^A = \mathbf{R}_B^A \cdot \mathbf{X}^B + \mathbf{T}_B^A. \tag{2.25}$$

Accordingly, the temporal calibration concerns the temporal relation between sensor clocks. First and foremost, a temporal offset between the sensor clocks must be determined. Second, a temporal drift can cause the time offset to change over time. This drift can be quantified by linear, quadratic or even higher order polynomial fits to the elapsed time. Third, variable-length delays (jitter) can occur if the data from the sensors is timestamped at a common host computer that does not run a real-time operating system [35].

In this work, we assume the sensors to include a clock to timestamp the data directly after acquisition, and can thus neglect jitter. Furthermore, we disregard clock drifts and focus on determining a relative temporal offset between sensor clocks. Here, the underlying assumption is that the offset is constant over a short period of time since the drift is negligible

for short experiments. Accordingly, in the following we introduce related work that focuses on determining the extrinsic transformation and/or the relative timestamp offset between sensors.

Generally, there exist several approaches to determine the spatiotemporal calibration parameters externally by using additional hardware. For example, the extrinsic transformation parameters could be determined by using a laser tracker to accurately measure the sensor housings and deduce the coordinate system origins from a technical drawing of the sensors. To synchronize sensor clocks, for example, the Network Time Protocol (NTP) [15] could be employed. However, these external solutions require additional knowledge about the sensor (e.g. coordinate system origins), external hardware (e.g. laser tracker, network architecture) and/or cooperation from the sensor (e.g. adjusting the clock according to the NTP algorithm). In this work, we aim to develop methods for generic sensors, and thus must assume these sensors to be black-box systems. Therefore, we can only establish a relation between the sensors by relying on the sensor data. Consequently, the calibration approaches must be adjusted to the data provided by the sensor and cannot be applied to arbitrary sensor combinations. Therefore, in the following, we present related work organized according to the specific sensor combinations.

2.5.1 Camera to IMU

This section references related work that deals with the spatiotemporal calibration between camera and IMU using solely sensor data. The existing approaches can be divided into two categories: offline and online. While offline approaches generally need a calibration target (e.g. a checkerboard) and are carried out in a laboratory, online approaches can be employed, for example, on a moving car to continuously estimate the calibration parameters. However, online approaches are generally less accurate.

One online approach to determine the time offset and the extrinsic calibration parameters is introduced by Li and Mourikis [36]. The authors propose to include the spatiotemporal calibration parameters into the state vector of an Extended Kalman Filter (EKF) that performs vision-aided inertial navigation. Consequently, they are able to continuously track the parameters over time, and can thus account for clock drift. Moreover, the EKF allows them to deduce the uncertainty of the estimates of the parameters.

Qin and Shen present another online approach that computes only the temporal offset between the data streams of the sensors [37]. In order to do that, they include the time offset as an additional optimization parameter while their main goal is to perform SLAM using a monocular visual-inertial system. The authors' idea is to define the velocity of an image feature on the image plane and to assume this velocity to be constant for short periods of time. Consequently, the position of an image feature can be formulated as a function depending on the time offset. Thus, the time offset can be included in the cost function that is iteratively optimized using Gauss-Newton methods.

To accurately determine both the time offset and the extrinsic calibration parameters, Kelly et al. propose an offline approach that requires a checkerboard as the calibration target [38]. The authors gather measurement data by rotating their visual-inertial system in front of the checkerboard. Subsequently, they align the resulting three-dimensional rotation curves of both

sensors by a variant of the Iterative Closest Point (ICP) algorithm. This alignment directly results in the desired spatiotemporal calibration parameters.

Another offline approach is presented by Furgale, Rehder et al. [39, 40]. Similar to the work previously presented, the authors calibrate their visual-inertial system in a laboratory and use a checkerboard as the calibration target. Again, the temporal offset and the extrinsic calibration parameters are computed simultaneously. However, they translate the problem into a maximum likelihood estimation by employing analytical basis functions which allows them to determine the parameters in a continuous batch optimization procedure.

2.5.2 Camera to laser scanner

Accordingly, this section references work on the calibration between camera and laser scanner using only sensor data. In contrast to the previous section, however, we focus on the extrinsic calibration and neglect the temporal calibration since we assume the sensors to be synchronized (cf. Section 1.1.2). This focus on the extrinsic calibration is in accordance with the literature we studied. Except for the work by Rehder et al. [41] and our own previously published approach [42], most approaches are formulated to only determine the extrinsic calibration parameters. Moreover, we do not present online methods in this section as we develop an offline approach ourselves and believe that calibrating the rigidly mounted sensor setup once before the actual experiment suffices.

Previously, we developed an approach to compute the timestamp offset between a camera and laser scanner [42]. Here, the main goal is to fuse information from both sensors to perform SLAM. However, it is evident that a timestamp offset leads to less accurate results. Thus, we define an optimization criterion that reflects the clarity of the map built by the SLAM algorithm. Subsequently, an iterative brute-force approach allows us to determine the timestamp offset in a maximum likelihood fashion assuming zero-mean sensor errors.

To our knowledge, Rehder et al. introduce the only approach to perform a full spatiotemporal calibration between camera and laser scanner [41]. They employ a checkerboard as the calibration target and wave the sensors in front of this target to generate measurement data that constrains both the time offset and the extrinsic calibration parameters. To formulate their calibration problem, the authors use a continuous state representation that allows them to perform a maximum likelihood estimation.

Existing approaches to find the extrinsic transformation between camera and laser scanner can be divided into two categories: target-less or target-based. Target-less approaches do not depend on a dedicated calibration target and are particularly useful to compute the extrinsic transformation during the operation of the robot [43, 44]. However, due to the lack of distinct features that can be identified in both camera and laser scan data, these approaches are generally less accurate or require manual operation by the user. Moreover, the extrinsic transformation can generally be assumed to be static, and thus it suffices to perform the calibration once before operation.

Target-based approaches differ in the calibration target and the features they identify on it. While there exist some approaches employing spheres [45] or even more specialized calibration targets such as a polygonal planar board [46], most approaches rely on a checkerboard due to its availability and further usability for intrinsic camera calibration. Zhang and Pless were the first

to use a checkerboard for the extrinsic calibration of a 2D laser scanner and a camera [47]. They find the plane parameters of the checkerboard using the camera and subsequently minimize the distance of laser scan points to this plane.

In 2005, this idea was extended by Unnikrishnan and Hebert to a 3D laser scanner by additionally computing the plane parameters from the laser scan point cloud [48]. As a prerequisite, their approach requires a checkerboard that is clearly visible both in the laser scan and in the camera image. Subsequently, the user is prompted to gather data from multiple checkerboard poses to constrain the extrinsic calibration parameters in all directions. Afterwards, their algorithm extracts the plane parameters of the checkerboard from the data of both sensors. Consequently, they feed the resulting nonlinear equations into an optimization algorithm that minimizes the difference in observed orientation of the checkerboard and the distance of 3D scan points to the plane observed by the camera.

Building on Unnikrishnan's approach, Zhou et al. propose to use additional features from the checkerboard besides the plane parameters [49]. In their work, they additionally extract the border lines of the checkerboard from the data of both sensors, which allows them to formulate additional constraints. Consequently, they are able to compute the full 6 DOF transformation between camera and laser scanner from one single checkerboard pose, whereas Unnikrishnan and Hebert require at least three distinct poses. In addition, the authors prove that it suffices to rotate the checkerboard in front of the sensor setup, while a translational movement does not constrain the extrinsic transformation parameters further.

Up until here, all presented approaches disregard the sensor uncertainties. In contrast, Zhou and Deng introduce the only calibration approach that takes the uncertainty of the plane parameters into account [50]. However, they only employ the uncertainty to weight the different checkerboard poses during the optimization and do not propagate it to the final result. Consequently, none of the established approaches are able to directly assess the accuracy of the computed extrinsic transformation. Instead, they employ ground truth information to compute the calibration error for specific test cases that cannot serve as a general statement for different sensors and calibration environments. Furthermore, due to the zero mean error assumption, systematic errors occurring, for example, for the distance measurements of the laser scanner, cannot be taken into account.

2.6 Visual-LiDAR odometry

In mobile robotics, the pose of a robot can be computed globally in a pre-defined coordinate system (e.g. World Geodetic System 1984 (WGS84)) or incrementally starting from the robot's current pose. The global localization requires either an external positioning system (e.g. GNSS) or a known map from which the robot can re-identify known features. However, GNSS can fail to provide accurate localization information if, for example, high buildings are present and accurate maps require the environment to have been explored previously. In these cases, the robot can only localize itself in a local coordinate system and simultaneously build its own map. This procedure is known as Simultaneous Localization and Mapping (SLAM). The first step of SLAM is to perform an incremental localization which is also known as dead reckoning or odometry. Afterwards, for example visual features can be incorporated into a map that

is subsequently used to improve the robot's localization. If camera images are employed to perform the dead reckoning, it is usually also denoted as *visual odometry*. If additionally data from a laser scanner is used, the incremental localization procedure is referred to as *visual-LiDAR odometry*.

In this work, we only tackle the first problem of SLAM - namely the dead reckoning. A lot of research has been conducted on the topic of visual odometry, which refers to the estimation of a robot's 3D motion from visual camera data alone. The basic idea of feature-based visual odometry is to find visual features in successive images, estimate depth of the features and then compute the rigid body transformation that aligns those successive image features. The most basic setup to conduct visual odometry is a monocular camera. Many approaches have been investigated in this research area [51], while PTAM [52], SVO [53] and ORB-SLAM [54] remain the most prominent ones. However, using only one camera, the depth/scale of features is hard to determine robustly and is "liable to drift over time", as Strasdat et al. point out [5].

Therefore, it is beneficial to add a second camera to perform stereo visual odometry as is done for SOFT-SLAM [55] and ROCC [56]. While improving the problem of scale drift, stereo vision strongly depends on a precise camera calibration. Instead of a second camera, it is possible to add an IMU to perform visual-inertial odometry [57]. Here, the IMU is employed to compute an initial estimate of the robot's translational movement which can then be used to account for the scale drift. Another option is to use a ToF camera (e.g. Microsoft Kinect for Windows v2) to measure depth directly [58]. However, ToF cameras are generally only suitable for indoor environments as their maximum range is limited and they are negatively impacted by sunlight [59].

To measure distance accurately in outdoor environments, Zhang et al. propose to directly fuse information from camera and laser scanner [60, 61]. Their algorithm projects the laser scan points onto the image plane, and subsequently determines distance information for each visual feature. In order to do that, the authors select the three closest scan points for each image feature, and are thus able to interpolate the depth linearly. Thereby, they acquire 3D visual features that can be re-identified over time. It remains to employ a nonlinear optimization algorithm to solve the rigid body transformation. Moreover, the authors incorporate visual features for which the depth could not be computed into their optimization algorithm by defining their reprojection error as the cost.

In subsequent publications Zhang et al. gradually improve their previously proposed approach. In 2015, the authors add a second step to their approach that is supposed to reduce the drift of their laser-visual odometry estimation algorithm [62]. In order to do that, they build a map consisting of so-called edge and planar points that are detected in the point cloud of the laser scanner. Subsequently, they extract edge and planar points from each new point cloud and match them with the existing map using a nonlinear optimization algorithm that minimizes the distance of each new edge point to an edge in the map and the distance of each new planar point to a plane in the map.

In 2017, Zhang et al. introduce another improvement of their method [63, 64]. They add an IMU to compute an initial estimate of the robot's motion which allows them to increase the robustness of their approach with regard to sensor failures (e.g. in low-light environments). Furthermore, the authors introduce the concept of keyframes into their laser-visual odometry.

This means that they refrain from estimating the motion from image to image, but rather localize themselves relative to a previously defined keyframe for several subsequent images. In this way they are able to reduce the drift of their algorithm.

Another approach to improve the initial idea of Zhang et al. is proposed by Graeter et al. [65]. In their work, the authors adjust the estimation of depth for visual features. Similar to Zhang et al., the authors first project the scan points onto the image plane. Afterwards, they select points within an rectangle of a predefined size around each visual feature and try to fit a plane through those scan points to determine the depth of the visual feature. Scan points that do not satisfy the local plane assumption are rejected. Moreover, they treat scan points on the ground plane separately as they possess valuable information.

2.7 Relation to this work

In the following, this section classifies the presented related work in the context of this work and outlines the shortcomings which we aim to resolve in this thesis.

In Section 2.1 we introduced the sensors that are employed during this work, detailed their operating principle and highlighted possible sources of error. In contrast to the established stochastic error models introduced here, this work introduces a new bounded error model for each sensor. Since this new error model is supposed to take all types of error into account, the underlying operating principle of each sensor has to be understood.

Accordingly, Section 2.2 introduced the basic concepts of stochastic error modeling. Due to the advantages already depicted in Section 1.1.1, this work employs an error model based on interval analysis for common problems in robotics that have been tackled previously assuming known stochastic error distributions.

Since this work introduces approaches to perform an extrinsic calibration between sensors and the dead reckoning of a robot in 3D, rotations must be treated in 3D accordingly. Thus, Section 2.3 detailed two different parametrizations to represent a rotation in 3D. As depicted in this section, both parametrizations exhibit different advantages, and are thus employed for different approaches in this work.

The PnP problem introduced in Section 2.4 is important in the context of the extrinsic calibration of a camera. As depicted in this section, there exist several approaches to solve this problem assuming a known stochastic error distribution for the camera. However, since this work introduces approaches to perform the extrinsic calibration under interval uncertainty, the PnP problem has to be solved under interval uncertainty as well. Thus, this work presents a new approach to tackle the PnP problem in a bounded error context. In contrast to the presented approaches, this work introduces a deterministic approach that does not depend on initial values since no optimization technique is employed. Furthermore, it exhibits all advantages that interval-based approaches generally have over stochastic ones (cf. Section 1.1.1) such as compatibility with systematic errors (e.g. due to the manufacturing accuracy of the checkerboard).

As the core of successful sensor fusion are accurate spatiotemporal calibration parameters, Section 2.5 first introduced the parameters that are important for this work and then presented established approaches to compute those parameters from solely sensor data. All presented

approaches have in common that they assume a known stochastic and zero-mean error distribution for sensor errors. Moreover, often the resulting accuracy of the computed calibration parameters is not considered or only determined empirically. Thus, this work introduces new interval-based approaches for the spatiotemporal calibration between camera, laser scanner and IMU. In contrast to the established methods, the calibration accuracy is a direct result of our approaches as the sensor errors are propagated consistently. Furthermore, our approaches only require known bounds for the sensor errors and are compatible with systematic errors.

After computing the spatiotemporal calibration parameters between camera, laser scanner and IMU, we can finally fuse information from all three sensors to perform dead reckoning of a mobile robot. We depicted established methods in Section 2.6. These methods exhibit several shortcomings. First, the sensor errors are assumed to be zero-mean which is not always the case as we will explain later. Second, most approaches refrain from modeling and propagating the sensor errors to the localization result which makes it impossible for the user to judge the accuracy of the robot's pose.

The approach closest to our work is that by Zhang et al. and all the following improvements that build on it. Here, we identified a third shortcoming. Since the accuracy of laser scan points and image features is neglected during the fusion of this information, the depth accuracy of the visual features cannot be quantified. However, the accuracy of this depth estimation is vastly different (e.g. for features on planes and for features on edges). Moreover, it poses valuable information as the accuracy can be used to weight the depth augmented features during the pose estimation. To overcome all these shortcomings, we introduce an interval-based visual-LiDAR odometry approach that propagates sensor errors to the localization result in a consistent manner.

3 Interval Analysis

This chapter introduces interval analysis - a different kind of mathematics developed to put guaranteed bounds on rounding or measurement errors. In contrast to traditional scalar mathematics, computations are performed on sets rather than scalar values.

With the advent of modern computers, scientists began to deal with the question of how to model the error arising due to floating point values. Since real numbers of infinite precision cannot be represented using a computer that is based on finite floating point calculations, an error is inevitably made. This prompted Ramon E. Moore to publish the first notable book on interval analysis in 1966 leading to a rise of popularity [66].

However, the concept of using bounds to provide an approximation of not exactly known values is presumably as old as mathematics itself. Already the well known Greek mathematician Archimedes calculated bounds to give a reasonably small enclosure for π [67]. By drawing two polygons, each with 96 sides, inside and outside of a circle, he concluded: $3\frac{10}{71} < \pi < 3\frac{1}{7}$.

Nevertheless, intervals are not only suitable to provide bounds on irrational numbers or floating point values. On the contrary, intervals can be used to describe any kind of physical uncertainties. In the case of robotics, this allows us to bound measurement errors and perform all following computations in a guaranteed way using the tools provided by interval analysis.

The following notions and definitions are freely taken from Luc Jaulin's book [9], which established a link between interval analysis and robotics, and Simon Rohou's PhD thesis [68], which extended the notions of intervals analysis to dynamical systems. To represent and compute with intervals in a computer, we employ the *IBEX* library [69], which is openly available.

3.1 Relation to stochastic error distributions

In the field of robotics, uncertainties are usually modeled using stochastic error distributions (cf. Section 2.2). To describe sensor errors, most of the time a normal (or Gaussian) distribution is used since it allows simple computations and error propagation due to the necessity of two parameters only - the mean value μ and the standard deviation σ. Its probability density function is

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \tag{3.1}$$

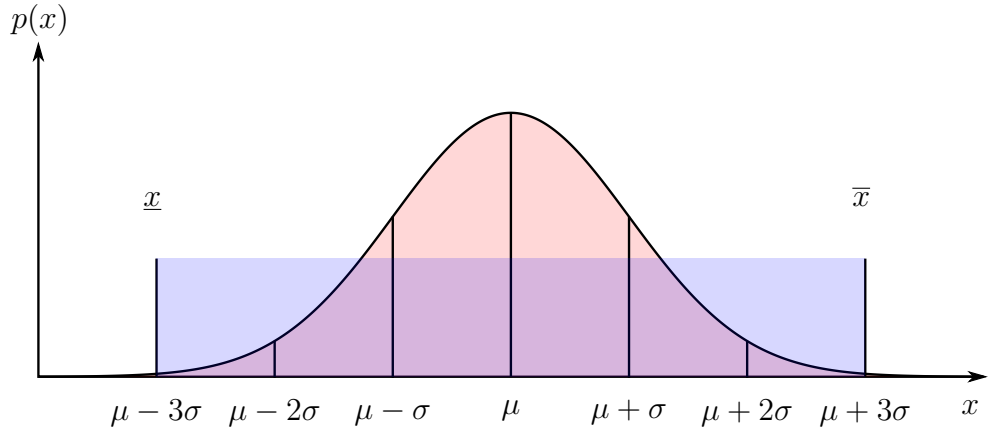


Figure 3.1: Exemplary relation between a Gaussian distribution and an interval. To guarantee a confidence rate of 99.73%, the interval bounds are set to $[x] = [\mu - 3\sigma, \mu + 3\sigma]$. Note that no assumption about the distribution between those bounds is made.

which can be used to provide a relative likelihood that the true value is x given the measurement μ and the standard deviation σ. As for any probability density function, the area under the curve has to amount to 1:

$$\int_{-\infty}^{\infty} p(x)\, dx = 1. \tag{3.2}$$

For the Gaussian distribution, it is possible to calculate the probability of measurements occurring in certain ranges around the expected value using the standard deviation. For example, the true value of a measurement μ is in the range of $\pm\sigma$ with a probability of 68.27%. Similarly, the range of $\pm 2\sigma$ corresponds to a probability of 95.45% and the range of $\pm 3\sigma$ corresponds to a probability of 99.73%.

Another stochastic distribution, which might intuitively appear similar to an interval, is the uniform distribution. Here, each value inside an interval $[a, b]$ is assigned the same probability. Thus, the probability density function is

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{if } x \in [a, b], \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

However, a uniform distribution is not the same as an interval, as the following example by Kreinovich illustrates [70]. Let us assume that we have n independent measurements with the same uncertainty that we want to add. According to the central limit theorem, the error distribution of the sum of n uniformly distributed measurements tends to a Gaussian distribution that grows with n as $\sqrt{n}$. In contrast, when modeling these n measurements with intervals, the error bounds of the sum grow with n as n. As a result, for large n, the error is seriously underestimated if a uniform distribution is chosen instead of an interval.

The main difference for all stochastic distributions to interval analysis is the assumption about knowing the precise distribution of measurements. While for both presented distributions each value has a specific probability of occurring, which can be computed according to the probability density functions, interval analysis only allows us to state if a value could be possible (i.e. it belongs to the interval), or if it is outside the interval. However, no statement about

the probability of each value is made, which means that the true distribution can be arbitrary without violating the quality of the interval error model.

Thus, comparing interval analysis to stochastic distributions means comparing two different ways of modeling sensor errors - each with different assumptions about the real world. While stochastic approaches assume the error distributions to be known, interval-based approaches consider the error bounds to be available and to reliably enclose the true value. Hence, we can argue that the assumption about knowing the exact error distribution is more strict than only assuming the error bounds to be known - without making any additional statement about the distribution between those bounds. However, in practice none of those assumptions is universally correct, and thus it depends on which model is more appropriate and which advantages (cf. Section 1.1.1) prevail.

Fig. 3.1 shows the relation between interval bounds and a Gaussian distribution. In the case that sensor data sheets provide us only with the standard deviation σ of a measurement μ, it is convenient to enclose the true value with a confidence rate of 99.73% by setting the interval bounds to $[x] = [\mu - 3\sigma, \mu + 3\sigma]$. Potential outliers can be taken care of by employing special interval analysis tools (cf. Section 3.9).

In summary, no direct connection exists between stochastic distributions and interval-based error modeling since fundamentally different assumptions are made. In the case of unknown error distributions, interval analysis prevails since it is compatible with any distribution. In particular, stochastic distributions suffer if the mean error deviates from zero while interval analysis is still applicable [71].

3.2 Basic notions

An interval $[x]$ is a closed and connected subset of $\mathbb{R}$. The set of all intervals is denoted by $\mathbb{IR}$. The interval $[x]$ is defined by its lower and upper bounds $\underline{x}$ and $\overline{x}$, respectively, such that

$$[x] = [\underline{x}, \overline{x}] = \{\, x \in \mathbb{R} \mid \underline{x} \leq x \leq \overline{x} \,\}. \tag{3.4}$$

Both, the lower and upper bounds can be infinite. In the case of $\underline{x} = \overline{x}$ the interval $[x]$ is said to be degenerate. Using this formalism, any real number can be considered as a degenerate interval. Finally, the empty interval - which represents the absence of a solution for our problems - is denoted by $\varnothing$.

The width of an interval $[x]$ is defined as

$$w([x]) = \overline{x} - \underline{x}. \tag{3.5}$$

It can be understood as the uncertainty on x^* which is the actual but unknown value we aim to estimate. Similarly, the radius of an interval $[x]$ is defined as

$$r([x]) = \frac{\overline{x} - \underline{x}}{2}. \tag{3.6}$$

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

$$\mathsf{mid}([x]) = \frac{\underline{x} + \overline{x}}{2}. \tag{3.7}$$

If we require a scalar output for our set-membership approach, the midpoint may serve as an intuitive approximation of x^*.

Example 3.2.1. The following are examples for intervals, their width and their midpoints:

$[x]$	$w([x])$	$\mathsf{mid}([x])$
$[1, 5]$	4	3
$[3]$	3	3
$\varnothing$	undefined	undefined

3.3 Set-theoretic operations

Given two intervals $[x]$ and $[y]$, we can compute their intersection as defined in set theory:

$$[x] \cap [y] = \{\, z \in \mathbb{R} \mid z \in [x] \text{ and } z \in [y] \,\}. \tag{3.8}$$

Similarly, their union is

$$[x] \cup [y] = \{\, z \in \mathbb{R} \mid z \in [x] \text{ or } z \in [y] \,\}. \tag{3.9}$$

While the intersection of two intervals results in a new interval, the union of two intervals is not necessarily an interval since the resulting set can be disconnected.

Example 3.3.1. $[1, 2] \cup [4, 5]$ is not an interval, since it results in a disconnected set. For example, the number 3 is missing in this set.

To overcome this problem, we use the interval hull of $[x] \cup [y]$ to create the smallest interval containing both $[x]$ and $[y]$:

$$[x] \sqcup [y] = [[x] \cup [y]], \tag{3.10}$$

where $[\{\mathbb{X}\}]$ is the smallest interval containing all values from the set $\mathbb{X}$.

Example 3.3.2. The following are examples for set-theoretic operations:

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

3.4 Interval computations

Up until here, only concepts of set theory have been introduced. However, the main idea of interval analysis is to extend the classical real arithmetic operators to set theory, making it possible to perform arithmetic computations with intervals/sets. Let $\diamond$ be one of the four

classical operators, i.e. $\diamond \in \{+, -, \cdot, /\}$. Given two intervals $[x]$ and $[y]$, performing $[x] \diamond [y]$ means computing the smallest interval that contains $x \diamond y$ for every $x \in [x]$ and $y \in [y]$:

$$[x] \diamond [y] = [\{\, x \diamond y \in \mathbb{R} \mid x \in [x],\, y \in [y] \,\}]. \tag{3.11}$$

Since we are dealing with closed intervals, the interval bounds are sufficient to perform the aforementioned operations, e.g.

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

Definitions for $\cdot$ and $/$ can be found in [9].

Example 3.4.1.

- $[-3, 1] + [4, 7] = [-3 + 4, 1 + 7] = [1, 8]$.
- $[2, 4] - [1, 6] = [2 - 6, 4 - 1] = [-4, 3]$.

However, some properties of the basic operators in $\mathbb{R}$ differ from those of their interval counterpart. For example, in general, $[x] - [x] \neq [0]$, because $[x] - [x] = [\underline{x} - \overline{x}, \overline{x} - \underline{x}]$. Furthermore, it holds that

$$[x] \cdot ([y] + [z]) \subset [x] \cdot [y] + [x] \cdot [z]. \tag{3.12}$$

Thus, it is sometimes important to simplify an equation as much as possible to create tighter intervals.

Further operators that have to be extended to interval arithmetic are elementary functions such as $\sin$, $\cos$, $\exp$. We give a general definition for these functions while taking into account that a discontinuous function should not result in a disconnected interval:

$$[f]([x]) = [\{\, f(x) \mid x \in [x] \,\}]. \tag{3.13}$$

For monotonic functions such as $\exp$ the evaluation is straightforward since monotonicity means that the function only increases or decreases for an increasing argument. Thus, evaluating the function at the bounds of the argument is sufficient to compute its image:

$$[\exp]([x]) = [\exp(\underline{x}), \exp(\overline{x})]. \tag{3.14}$$

However, for non-monotonic functions the computation is more complicated. For example, using only the bounds to compute $[\cos]([-\pi, \pi])$ results in $[\cos(-\pi), \cos(\pi)] = [-1, -1]$, which differs from the correct result $[\cos]([-\pi, \pi]) = [-1, 1]$. To compute these kind of functions, special algorithms have been built [9].

3.5 Interval vectors

An interval vector (also called box) is the Cartesian product of n intervals, and thus a subset of $\mathbb{R}^n$:

$$[\mathbf{x}] = [x_1] \times [x_2] \times \cdots \times [x_n], \tag{3.15}$$

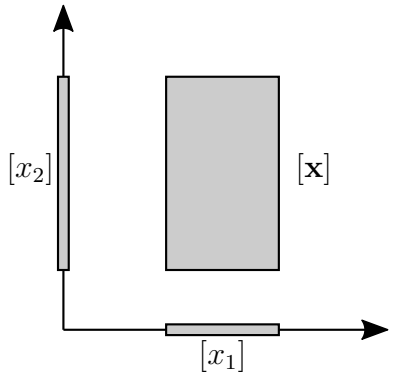


Figure 3.2: Example for an interval vector $[\mathbf{x}] = [x_1] \times [x_2]$ and the projections onto the axes.

where $[x_i] = [\underline{x_i}, \overline{x_i}]$, for $i = 1, \ldots, n$.

An interval vector is also denoted as $[\mathbf{x}] = (\,[x_1] \;\; \ldots \;\; [x_n]\,)^\intercal$. The set of all interval vectors is denoted as $\mathbb{IR}^n$. Naturally, an interval vector $[\mathbf{x}]$ is an axis-aligned box, and thus it's i-th component is the projection of $[\mathbf{x}]$ onto the i-th axis. An example for $[\mathbf{x}] \in \mathbb{IR}^2$ is depicted in Fig. 3.2.

The basic operations on intervals - such as finding the lower bound, upper bound, midpoint or width (cf. Section 3.2) - can be easily extended to interval vectors by performing the corresponding operation on each component of the interval vector. For example, the midpoint of an interval vector $[\mathbf{x}]$ is

$$\mathsf{mid}([\mathbf{x}]) = (\mathsf{mid}([x_1]), \ldots, \mathsf{mid}([x_n]))^\intercal. \tag{3.16}$$

Naturally, the definitions introduced for interval vectors can be transferred to interval matrices accordingly.

3.6 Inclusion functions

Given a function $\mathbf{f}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$, it is possible to compute the image set $\mathbf{f}([\mathbf{x}])$ that results from applying the function $\mathbf{f}$ to every possible value of an interval vector $[\mathbf{x}]$. However, this image set may have any shape and is not necessarily a box. For instance, this set may consist of non-connected subsets and can have holes. An example for such a function $\mathbf{f}$ resulting in this kind of image set is depicted in Fig. 3.3. Since dealing with such sets is more complicated and can become computationally expensive, we strive for a box that is guaranteed to contain the set, i.e. we want to enclose the set in a box.

In order to do that, we define an interval function $[\mathbf{f}]$ from $\mathbb{IR}^n$ to $\mathbb{IR}^m$ as the inclusion function for $\mathbf{f}$ if it computes a box enclosing the image set for any input:

$$\forall\, [\mathbf{x}] \in \mathbb{IR}^n : \; \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]). \tag{3.17}$$

The purpose of inclusion functions is to compute a tight enclosure in a short time. A naive inclusion function for any function f is $[\mathbf{f}]([\mathbf{x}]) = \mathbb{R}^m$. While this inclusion function can be

computed quickly, the resulting box $[\mathbf{f}]([\mathbf{x}])$ is very large, and thus it is not a useful inclusion function. Fig. 3.3 shows an example of an inclusion function.

An inclusion function is said to be

- minimal if $\forall[\mathbf{x}]$, $[\mathbf{f}]([\mathbf{x}])$ is the smallest box containing $\mathbf{f}([\mathbf{x}])$. The unique, minimal inclusion function for $\mathbf{f}$ is denoted as $[\mathbf{f}]^*$ (cf. Fig. 3.3).
- thin if for any punctual interval vector the image is also degenerate: $[\mathbf{x}] = \mathbf{x}$, $[\mathbf{f}]([\mathbf{x}]) = \mathbf{f}(\mathbf{x})$.
- inclusion monotonic if the following holds: $[\mathbf{x}] \subset [\mathbf{y}] \Rightarrow [\mathbf{f}]([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{y}])$.

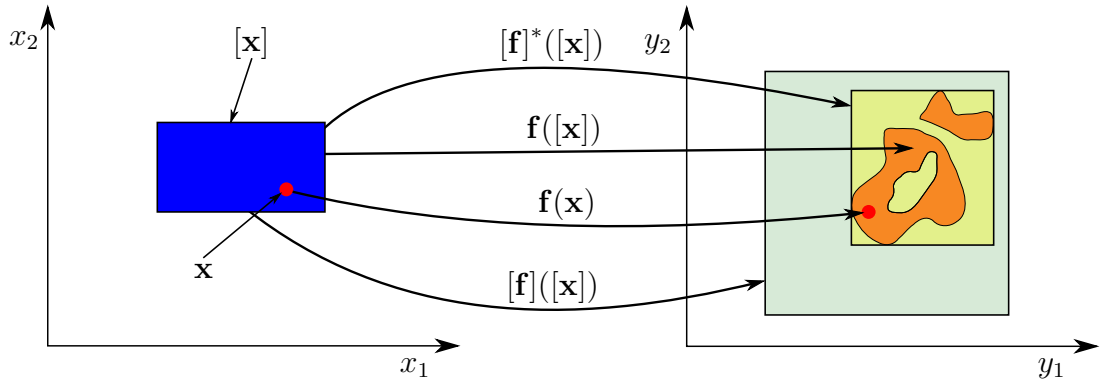


Figure 3.3: Examples for: a function $\mathbf{f}$ resulting in a disconnected image set $\mathbf{f}([\mathbf{x}])$; an inclusion function $[\mathbf{f}]$ resulting in the box $[\mathbf{f}]([\mathbf{x}])$ enclosing the image set; a minimal inclusion function $[\mathbf{f}]^*$ resulting in the smallest possible enclosure $[\mathbf{f}]^*([\mathbf{x}])$ of the image set.

3.6.1 Natural inclusion functions

A straightforward way to obtain an inclusion function for a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ that is composed of a finite number of the four classical operators $(+, -, \cdot, /)$ and elementary functions $(\sin, \cos, \tan, \sqrt{\cdot}, \ldots)$ is to replace each real variable x_i by an interval variable $[x_i]$ and each operator by the corresponding interval operator. An inclusion function $[\mathbf{f}]$ obtained this way is called a *natural inclusion function*.

Due to design, each natural inclusion function is thin and inclusion monotonic. However, in general it is not minimal because of dependencies between variables (cf. Section 3.4). Nevertheless, if it is composed of continuous operators and functions only, and each of the variables $[x_i]$ appears only once, then $[\mathbf{f}]$ is minimal.

Example 3.6.1. Consider the natural inclusion function $f_1(x) = 2x - x$. It is not minimal due to dependencies between the variables. Reformulating the expression to an equivalent function $f_2(x) = x$ yields a minimal inclusion function since the variable x appears only once. Evaluating both functions for $[x] = [1, 2]$ shows this:

$$[f_1]([x]) = 2 \cdot [1,2] - [1,2] = [2,4] - [1,2] = [0,3] \supset [1,2] = [f_2]([x]). \tag{3.18}$$

3.7 Contractors

Starting from an initial domain, we aim to compute the smallest domain for variables that satisfy a set of constraints. However, characterizing the solution set $\mathbb{S}$ is NP-hard in general. Thus, we introduce *contractors* in the following section which often allow to compute a reasonably tight enclosure for the solution set while keeping the computation time polynomial. First, however, we present the formulation of a CSP for which contractors can be built.

3.7.1 Constraint Satisfaction Problem (CSP)

Let $x_i,\ 1 \leq i \leq n_x$, be a set of variables and $f_j,\ 1 \leq j \leq n_f$, a set of arbitrary functions that link those variables as follows

$$f_j(x_1, x_2, \ldots, x_{n_x}) = 0. \tag{3.19}$$

Here, the functions act as constraints on the variables and may allow us to reduce the width of each variables' initial domain. Let

$$\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \ldots & x_{n_x} \end{pmatrix}^\mathsf{T} \tag{3.20}$$

be the variable vector and

$$[\mathbf{x}] = [x_1] \times [x_2] \times \cdots \times [x_{n_x}] \tag{3.21}$$

be the initial domain for those boxes (i.e. $x_i \in [x_i],\ 1 \leq i \leq n_x$). Furthermore, $\mathbf{f}$ is the vector function consisting of all f_j. Now, summarizing (3.19) yields $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. This allows us to formulate the *CSP* $\mathcal{H}$ as

$$\mathcal{H}:\ (\mathbf{f}(\mathbf{x}) = \mathbf{0},\ \mathbf{x} \in [\mathbf{x}]). \tag{3.22}$$

A possible solution to $\mathcal{H}$ is a mapping that assigns a value from its domain to each variable such that all constraints are satisfied. Thus, the solution set $\mathbb{S}$ of $\mathcal{H}$ is

$$\mathbb{S} = \{\, \mathbf{x} \in [\mathbf{x}] \mid \mathbf{f}(\mathbf{x}) = \mathbf{0} \,\}. \tag{3.23}$$

Example 3.7.1. An exemplary CSP is the following set of constraints and initial domains:

$$\mathcal{H}: \begin{pmatrix} \mathbf{f}(\mathbf{x}) = \begin{cases} x_2 - \exp(x_1) = 0 \\ \sin(x_1) + x_3 = 0 \end{cases} \\ [x_1] = [2,3],\ [x_2] = [0,1],\ [x_3] = [-1, 0.5] \end{pmatrix}. \tag{3.24}$$

3.7.2 Contractors

To approximate the solution set of a CSP, it is possible to *contract* the variables' initial domains. In this case, contracting means replacing the (initially large) domain $[\mathbf{x}]$ by a smaller domain $[\mathbf{x}'] \subset [\mathbf{x}]$ such that $\mathbb{S} \subset [\mathbf{x}']$. In other words, only parts that are not contained in the solution set $\mathbb{S}$ are removed from $[\mathbf{x}]$. Any operator that can perform such a contraction is denoted as a *contractor* $\mathcal{C}$. However, to avoid an exponential time complexity, contractors cannot bisect the

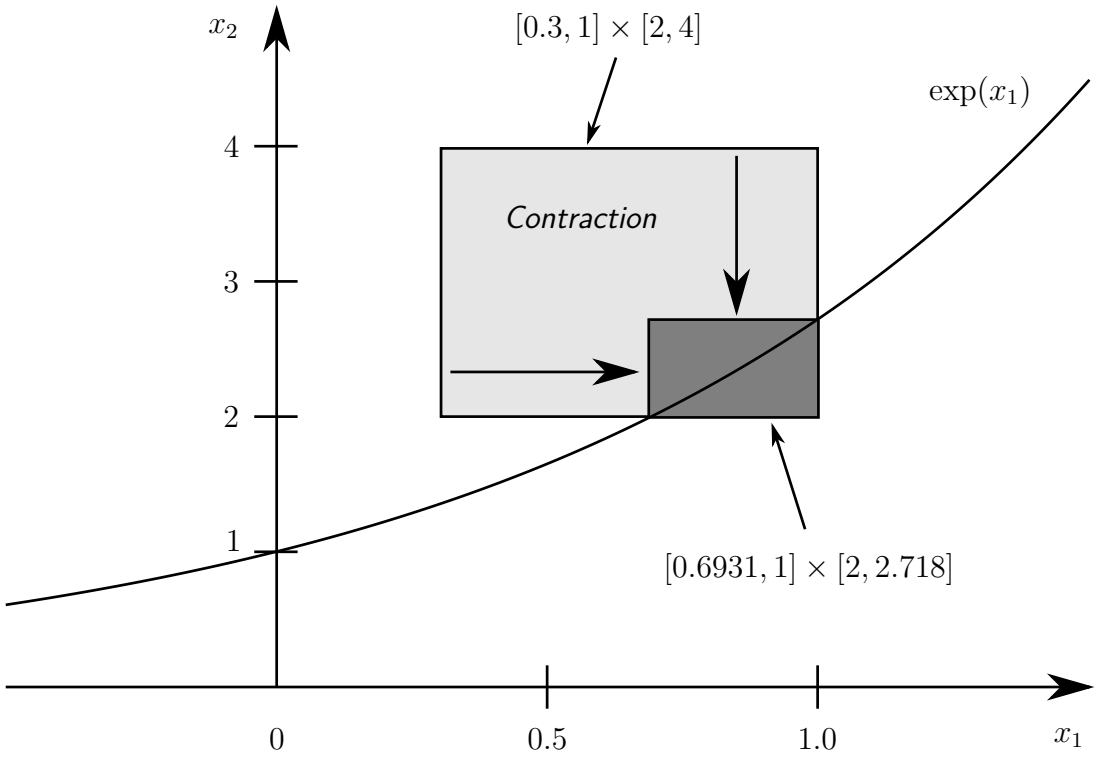


Figure 3.4: Illustration of how a contractor works. An initially large box $[0.3, 1] \times [2, 4]$ is contracted to the minimal box $[0.6931, 1] \times [2, 2.718]$ to better approximate the exponential function without losing any part of the solution. Example 3.7.2 shows the corresponding steps in the algorithm.

domain (in contrast to the SIVIA algorithm, which will be explained in Section 3.8). Fig. 3.4 shows the general idea of a contractor.

Formally defined, a contractor is a mapping $\mathcal{C}$ from $\mathbb{IR}^n$ to $\mathbb{IR}^n$ such that the following properties hold [72]:

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

Furthermore, we denote a box as *insensitive* to $\mathcal{C}$ if $\mathcal{C}([\mathbf{x}]) = [\mathbf{x}]$. Otherwise, it is said to be *sensitive*. The first property ensures that a contractor can only reduce a box, and not enlarge it. The second property guarantees that no insensitive points will be removed. Together, these properties ensure that a contractor never loses a part of the solution and can be applied as many times as desired since the box can only get smaller.

Other important properties of a contractor $\mathcal{C}$ are:

3. **Continuity**: $\mathcal{C}(\{\mathbf{x}\}) = \varnothing \Leftrightarrow (\exists \epsilon > 0, \forall [\mathbf{x}] \subseteq B(\mathbf{x}, \epsilon), \mathcal{C}([\mathbf{x}]) = \varnothing)$, where $B(\mathbf{x}, \epsilon)$ is the ball centered on $\mathbf{x}$ with radius ϵ.
4. **Monotonicity**: $[\mathbf{x}] \subseteq [\mathbf{y}] \Rightarrow \mathcal{C}([\mathbf{x}]) \subseteq \mathcal{C}([\mathbf{y}])$.
5. **Minimality**: $\forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathcal{C}([\mathbf{x}]) = [[\mathbf{x}] \cap \mathsf{set}(\mathcal{C})]$.
6. **Idempotence**: $\forall [\mathbf{x}] \in \mathbb{IR}^n,\ \mathcal{C}(\mathcal{C}([\mathbf{x}])) = \mathcal{C}([\mathbf{x}])$.

Especially the fifth property is important since we are often interested in finding the *minimal* contractor that manages to contract the box $[\mathbf{x}]$ to the smallest box containing the solution set. Moreover, the third property enforces that the set

$$\mathsf{set}(\mathcal{C}) = \{\, \mathbf{x} \in \mathbb{R}^n \mid \mathcal{C}(\mathbf{x}) = \mathbf{x} \,\} \tag{3.25}$$

is closed. This is required to prove the convergence of contractors.

One of the most important contractors we want to introduce here is the forward-backward contractor [73]. It decomposes all constraints of a CSP $\mathcal{H} : \mathbf{f}(\mathbf{x}) = 0,\ \mathbf{x} \in [\mathbf{x}]$ into primitive constraints and considers them in isolation. In contrast to many different contractors, of which some can be found in [9], the number of constraints n_f is not necessarily equal to the number of variables n_x. The following example illustrates the operations of the forward-backward contractor.

Example 3.7.2. Consider a reduced version of the CSP given in Example 3.7.1:

$$\mathcal{H} : \begin{pmatrix} f(\mathbf{x}) = x_2 - \exp(x_1) = 0 \\ [x_1] = [0.3, 1],\ [x_2] = [2, 4] \end{pmatrix} \tag{3.26}$$

For the forward propagation, the constraints are split into a finite sequence of elementary operations to compute $y = f(\mathbf{x})$. Intermediate variables $a_i,\ i > 0$ are used to express these operations:

$$\begin{aligned} [a_1] &\coloneqq \exp([x_1]) \quad &\Rightarrow \quad [a_1] &= \exp([0.3, 1]) = [1.350, 2.718], \\ [y] &\coloneqq [x_2] - [a_1] \quad &\Rightarrow \quad [y] &= [2, 4] - [1.350, 2.718] = [-0.718, 2.650]. \end{aligned}$$

$[y]$ is taken equal to $\{0\}$ since $f(\mathbf{x}) = 0$:

$$[y] \coloneqq [y] \cap \{0\} \quad \Rightarrow \quad [y] = \{0\}.$$

If $[y]$ turns out to be empty, the CSP has no solution. Subsequently, the backward propagation is performed by computing all possible reformulations of the elementary operations:

$$\begin{aligned} [x_2] &\coloneqq [x_2] \cap ([y] + [a_1]) \quad &\Rightarrow \quad [x_2] &= [2, 4] \cap ([0] + [1.350, 2.718]) = [2, 2.718], \\ [a_1] &\coloneqq [a_1] \cap ([x_2] - [y]) \quad &\Rightarrow \quad [a_1] &= [1.350, 2.718] \cap ([2, 2.718] - [0]) = [2, 2.718], \\ [x_1] &\coloneqq [x_1] \cap \log([a_1]) \quad &\Rightarrow \quad [x_1] &= [0.3, 1] \cap \log([2, 2.718]) = [0.6931, 1]. \end{aligned}$$

Finally, $[\mathbf{x}] = [0.6931, 1] \times [2, 2.718]$ is the contracted (and in this case minimal) domain for the CSP $\mathcal{H}$. Fig. 3.4 visualizes the contraction performed in this example.

Generally, the forward-backward contractor is not minimal and must be applied numerous times in succession to find a reasonably small box. It is minimal, however, if each variable appears only once in all constraints together.

3.8 Set Inversion Via Interval Analysis (SIVIA)

Often, non-linear functions have to be inverted in mobile robotics to compute the solution to various problems. For example, given a distance function and the corresponding distances to landmarks, we can compute the robot's position in 2D by inverting the distance function and intersecting the resulting circles. However, inverting a non-linear function is not straightforward. To do this in a bounded-error context, SIVIA was introduced by Jaulin and Walter [74]. In the following, this section introduces the corresponding algorithm, but first explains the wrapping effect and introduces subpavings, which are the result of SIVIA.

3.8.1 Wrapping effect

Since intervals and interval vectors are axis-aligned, we introduce pessimism whenever we try to enclose a set that is not an axis-aligned box. This phenomenon is called *wrapping effect*. If we evaluate a function which suffers from the wrapping effect multiple times in a row, this pessimism can quickly increase. Moore [66] provides an intuitive example for this problem by applying consecutive rotations to a box. Fig. 3.5 shows this example. To overcome the wrapping effect, we introduce a set of non-overlapping boxes to describe the solution set in the following section.

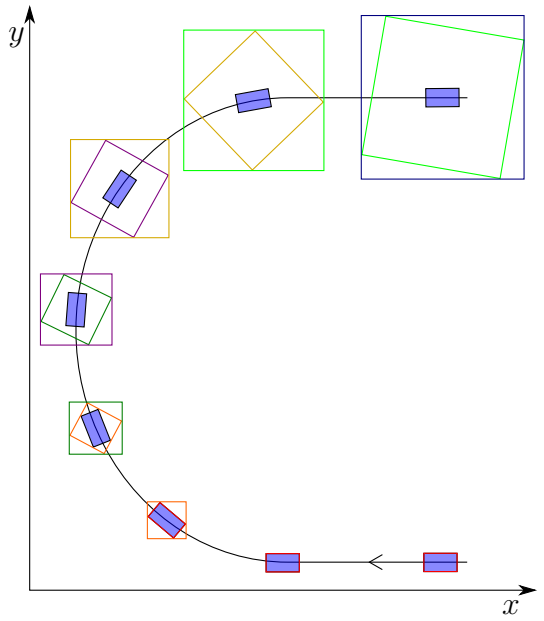
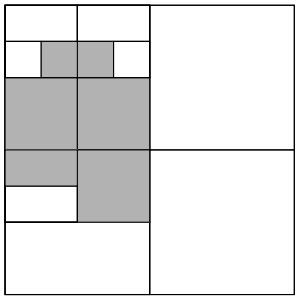
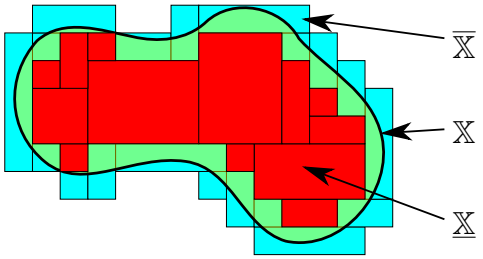


Figure 3.5: Illustration of the wrapping effect for a robot taking a 180° turn. Starting in the bottom right corner, the robot (blue box) moves along the black path. At first, the interval box (red) enclosing the robot is very small since the robot's position is known accurately. However, after starting to turn, we compute the robot's rotation and apply the very same rotation to the interval box. Now, we need to again find an axis aligned interval box enclosing the robot's position, which is depicted in orange. After continuing further on the path, the new interval box is again rotated according to the robot's rotation and enclosed using a new axis aligned box (green). Finally, after some more identical steps, the interval box enclosing the robot has become very large (blue).



(a) Visualization of bisections. The gray boxes form a regular subpaving.



(b) Illustration of an inner subpaving $\underline{\mathbb{X}}$ (red) and outer subpaving $\overline{\mathbb{X}}$ (blue) of the solution set $\mathbb{X}$ (green).

Figure 3.6: Visualization of regular subpavings.

3.8.2 Subpavings

Let $\mathbb{K}$ be a set of n boxes:

$$\mathbb{K} = \{[\mathbf{x_1}], [\mathbf{x_2}], \ldots, [\mathbf{x_n}]\}. \tag{3.27}$$

$\mathbb{K}$ is called a *subpaving* if the boxes x_i, $1 \leq i \leq n$ are non-overlapping.

Let $[\mathbf{x}]$ be a box enclosing the solution set which we want to approximate more accurately using a subpaving. To do this, we use a finite number of bisections and selections on $[\mathbf{x}]$ (i.e. recursively splitting $[\mathbf{x}]$ in half and keeping only those boxes that contain parts of the solution set). The set of boxes $\mathbb{K}$ resulting from this iterative process is called a *regular* subpaving. Fig. 3.6a shows a regular subpaving and its origin from multiple bisections. Typically, bisections are employed to halve the boxes in their widest dimension.

There exist two different types of subpavings to describe a solution set $\mathbb{X}$. The inner subpaving $\underline{\mathbb{X}}$ consists only of boxes that lie completely in the solution set, while the outer subpaving $\overline{\mathbb{X}}$ contains the inner subpaving and additionally all boxes that contain parts of the solution set. Thus,

$$\underline{\mathbb{X}} \subseteq \mathbb{X} \subseteq \overline{\mathbb{X}} \tag{3.28}$$

Fig. 3.6b shows an example for an inner and an outer subpaving. Usually, we are interested in computing an enclosure for the solution set, and thus we will compute an outer subpaving.

3.8.3 SIVIA algorithm

Set inversion is an operation to characterize the set

$$\mathbb{X} = \{\, \mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y} \,\} = \mathbf{f}^{-1}(\mathbb{Y}), \tag{3.29}$$

where $\mathbf{f}$ is a possibly non-linear function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbb{Y}$ is a subset of $\mathbb{R}^m$. The SIVIA algorithm [74] allows to compute an approximation for $\mathbb{X}$ for any $\mathbb{Y}$ and any function $\mathbf{f}$ for which a convergent inclusion function $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^m$ exists. As introduced in the previous section, SIVIA allows to compute an inner and outer subpaving enclosing the solution set (cf. (3.28)). Fig. 3.7 shows the general idea of SIVIA.

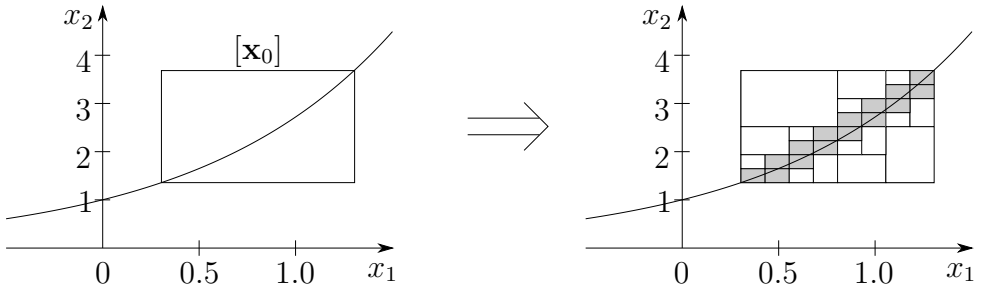


Figure 3.7: Illustration of how SIVIA works. An initially large box $[\mathbf{x}_0]$ is bisected to better approximate the exponential function. Gray boxes contain part of the solution while white boxes are discarded. Note that $[\mathbf{x}_0]$ cannot be contracted using a contractor since it is already the minimal box enclosing the function.

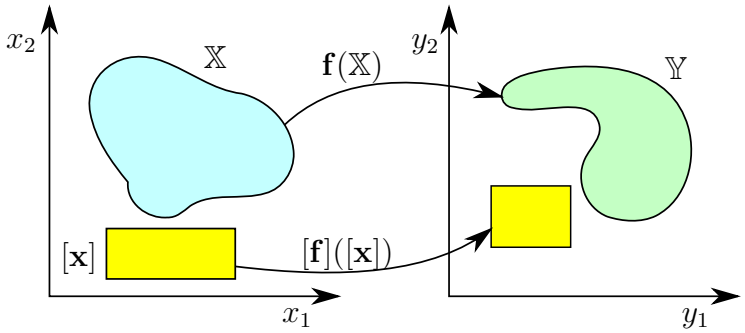
Algorithm 1: SIVIA

input : $[\mathbf{f}]$, $[\mathbf{x}_0]$, $\mathbb{Y}$, ϵ
output : $\underline{\mathbb{X}}$, $\overline{\mathbb{X}}$

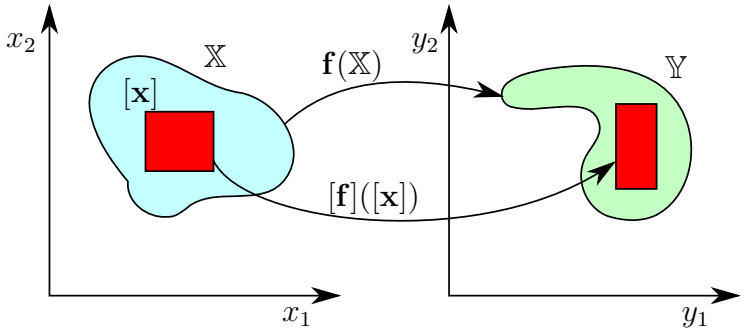
$\mathcal{S} \coloneqq \{[\mathbf{x}_0]\}$; // $\mathcal{S}$ is a stack
while $\mathcal{S} \neq \varnothing$ **do**
  $[\mathbf{x}] \coloneqq \mathsf{top}(\mathcal{S})$;
  if $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \varnothing$ **then**
    continue;
  else if $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$ **then**
    $\overline{\mathbb{X}} \coloneqq \overline{\mathbb{X}} \cup [\mathbf{x}]$;
    $\underline{\mathbb{X}} \coloneqq \underline{\mathbb{X}} \cup [\mathbf{x}]$;
  else if $w([\mathbf{x}]) < \epsilon$ **then**
    $\overline{\mathbb{X}} \coloneqq \overline{\mathbb{X}} \cup [\mathbf{x}]$;
  else
    $([\mathbf{x}_1], [\mathbf{x}_2]) \coloneqq \mathsf{bisect}([\mathbf{x}])$;
    $\mathcal{S} \coloneqq \mathcal{S} \cup \{[\mathbf{x}_1]\} \cup \{[\mathbf{x}_2]\}$;
  end
end

In the beginning, SIVIA requires an arbitrarily large box $[\mathbf{x}_0]$ that is guaranteed to contain the outer subpaving $\overline{\mathbb{X}}$. Starting from this box, SIVIA will check if it belongs to $\overline{\mathbb{X}}$, both $\underline{\mathbb{X}}$ and $\overline{\mathbb{X}}$, or if it is not part of the solution set. If the box is too large to decide, SIVIA will bisect the box and continue with the two sub-boxes recursively. An exemplary version of the algorithm is depicted in Algorithm 1. The four different cases encountered in the algorithm can be characterized as follows and are illustrated in Fig. 3.8:

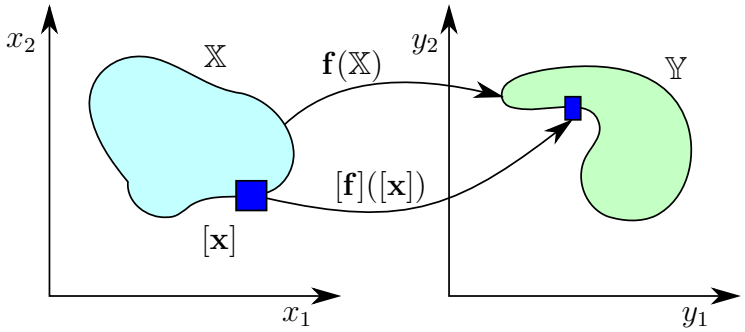
1. $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \varnothing$: $[\mathbf{x}]$ does not lie in the solution set, and thus it is not further considered.
2. $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$: $[\mathbf{x}]$ belongs completely to the solution set, and thus we can add it to both the inner and outer subpaving.
3. $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} \neq \varnothing$ and $w([\mathbf{x}]) < \epsilon$: At least some part of $[\mathbf{x}]$ belongs to the solution set. Since the box is already small enough with respect to the algorithm's expected precision, we stop the computation on this box and add it to the outer subpaving.
4. Otherwise, we bisect $[\mathbf{x}]$ (usually along it's widest dimension) and continue the computation with the resulting sub-boxes.



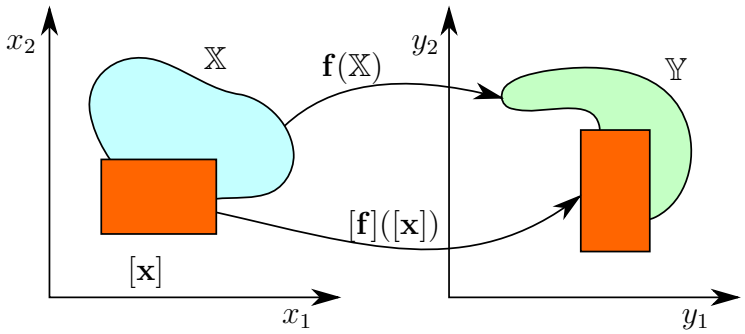
(a) First case: $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \varnothing$: $[\mathbf{x}]$ does not belong to the solution set.



(b) Second case: $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$: $[\mathbf{x}]$ belongs completely to the solution set.



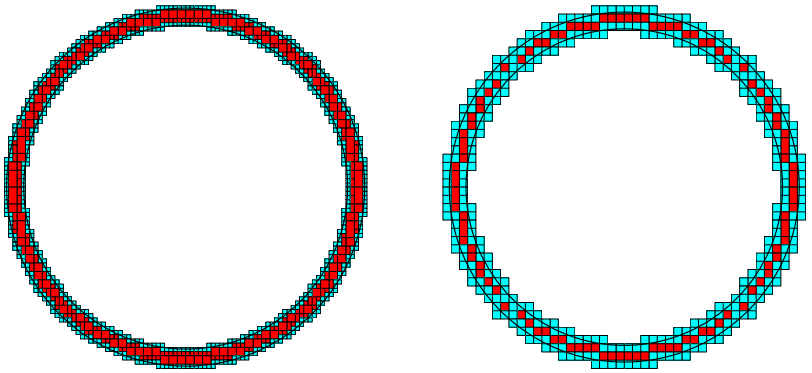
(c) Third case: $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} \neq \varnothing$ and $w([\mathbf{x}]) < \epsilon$: At least some part of $[\mathbf{x}]$ belongs to the solution set.



(d) Fourth case: $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} \neq \varnothing$ and $w([\mathbf{x}]) > \epsilon$: $[\mathbf{x}]$ is undetermined and will be bisected.

Figure 3.8: Illustration of the four cases encountered by the SIVIA algorithm. The set on the left in light blue is the set to be computed: $\mathbb{X} = \mathbf{f}^{-1}(\mathbb{Y})$.

The only parameter of SIVIA is ϵ, which is used to set the precision of the approximation of the solution set. The smaller we choose ϵ, the more bisections occur and the smaller are the boxes in the outer subpaving. However, this comes at the cost of an increased computation time. Fig. 3.9 shows examples for subpavings of different accuracies as computed by SIVIA.



(a) Small $\epsilon \rightarrow$ high accuracy. (b) Large $\epsilon \rightarrow$ low accuracy.

Figure 3.9: Subpavings of different accuracy levels computed by the SIVIA algorithm to enclose a circle with an unknown but bounded radius. The boundary of the true solution set is depicted in black. The inner and outer subpavings are colored in red and blue, respectively.

There exist different ways to reduce the computation time of SIVIA or to increase the accuracy. For example, SIVIA can be parallelized to allow the simultaneous evaluation of different boxes from the stack. Furthermore, SIVIA can be used in conjunction with contractors. This allows to omit some bisections by contracting the boxes beforehand. However, this comes at the price that the resulting subpaving may no longer be regular. Algorithm 2 shows a simple version of the SIVIA algorithm which bases its computations on the contractor $\mathcal{C}$. In contrast to the previous algorithm, it computes only the outer subpaving. However, for most applications which require an outer enclosure of the solution set only, this is sufficient.

Algorithm 2: SIVIA with contractor

input : $\mathcal{C}$, $[\mathbf{x}_0]$, ϵ
output : $\overline{\mathbb{X}}$

$\mathcal{S} \coloneqq \{[\mathbf{x}_0]\}$; // $\mathcal{S}$ is a stack
while $\mathcal{S} \neq \varnothing$ **do**
  $[\mathbf{x}] \coloneqq \mathsf{top}(\mathcal{S})$;
  $[\mathbf{x}] \coloneqq \mathcal{C}([\mathbf{x}])$;
  if $[\mathbf{x}] = \varnothing$ **then**
    continue;
  else if $w([\mathbf{x}]) < \epsilon$ **then**
    $\overline{\mathbb{X}} \coloneqq \overline{\mathbb{X}} \cup [\mathbf{x}]$;
  else
    $([\mathbf{x}_1], [\mathbf{x}_2]) \coloneqq \mathsf{bisect}([\mathbf{x}])$;
    $\mathcal{S} \coloneqq \mathcal{S} \cup \{[\mathbf{x}_1]\} \cup \{[\mathbf{x}_2]\}$;
  end
end

3.9 Relaxed intersection

In the case of outliers, the intersection of multiple intervals may be empty.

Example 3.9.1. Consider 5 intervals $[x_1] = [1, 3]$, $[x_2] = [2, 3]$, $[x_3] = [2, 5]$, $[x_4] = [4, 6]$, $[x_5] = [1, 5]$. Apparently, their intersection is empty:

$$[x_1] \cap [x_2] \cap [x_3] \cap [x_4] \cap [x_5] = \varnothing. \tag{3.30}$$

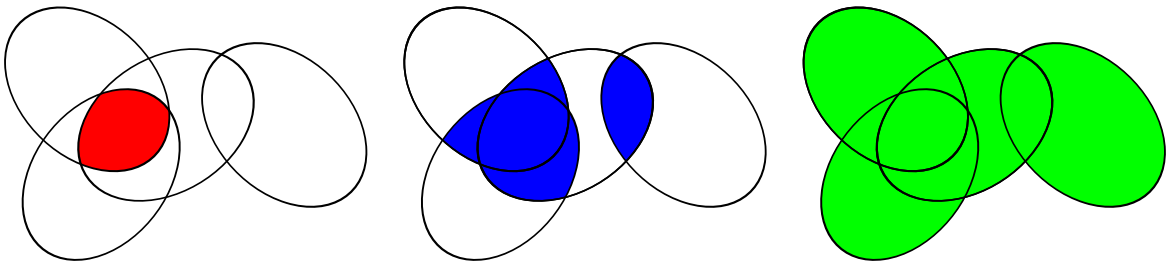
To allow for outliers, it is possible to compute a q-*relaxed intersection* meaning that a maximum of q outliers are tolerated and all other intervals have to overlap [75, 76]. This requires the additional assumption that we encounter q outliers *at most*. Given n intervals $[x_i]$, $1 \leq i \leq n$, we denote the q-relaxed intersection by $\bigcap_{1 \leq i \leq n}^{\{q\}} [\mathbf{x}_i]$. An illustration of the q-relaxed intersection is given in Fig. 3.10.

Example 3.9.2. The 1-relaxed intersection for Example 3.9.1 is

$$\bigcap_{1 \leq i \leq 5}^{\{1\}} [\mathbf{x}_i] = [2, 3]. \tag{3.31}$$

Furthermore, the 2-relaxed intersection is

$$\bigcap_{1 \leq i \leq 5}^{\{2\}} [\mathbf{x}_i] = [2, 5]. \tag{3.32}$$



(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 3.10: Exemplary q-relaxed intersection of four sets for $q \in \{1, 2, 3\}$.

3.10 Tubes

In the context of mobile robotics, sensors are often not synchronized and the robot may move between different measurement acquisitions. However, all techniques described up until here deal with *static* state estimation problems and have no means to take time uncertainties into account. In order to tackle this problem, Bethencourt and Jaulin introduced the notion of *tubes* [77] to extend trajectories to the set-membership field. Similar to [68], we denote a

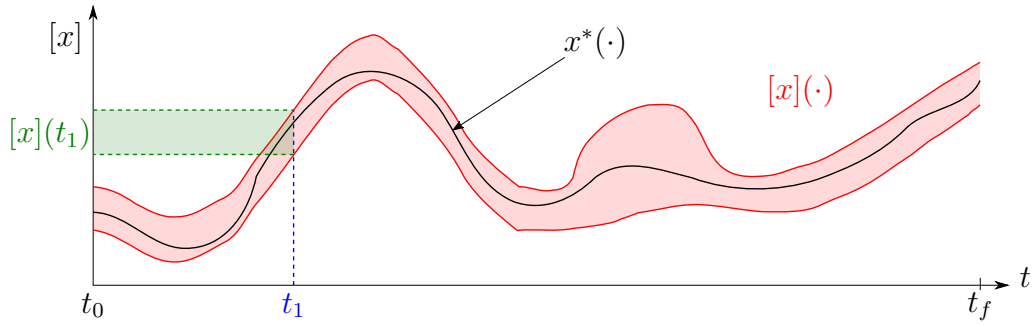


Figure 3.11: An exemplary tube $[x](\cdot)$ enclosing the true one-dimensional trajectory $x^*(\cdot)$. Selecting a distinct point in time t_1 results in an interval $[x](t_1)$ enclosing the true value $x^*(t_1)$.

single n-dimensional trajectory by $\mathbf{x}(\cdot) : \mathbb{R} \to \mathbb{R}^n$. The $(\cdot)$ notation is used to clearly separate the notion of a trajectory from a single evaluation of it: $\mathbf{x}(t) \in \mathbb{R}^n$.

A tube is defined as a set of trajectories over a time interval $[t] = [t_0, t_f]$. Consequently, it is denoted as $[\mathbf{x}](\cdot) : \mathbb{R} \to \mathbb{IR}^n$. Formally, a tube is defined by two trajectories, which depict the lower and upper bounds of the tube: $[\mathbf{x}](\cdot) = [\underline{\mathbf{x}}(\cdot), \overline{\mathbf{x}}(\cdot)]$, such that $\forall t \in [t] : \underline{x_i}(t) \leq \overline{x_i}(t)$, where $i = 1, \ldots, n$. Fig. 3.11 shows an exemplary one-dimensional tube. As can be seen, it encloses the true trajectory $x^*(\cdot)$. Generally, a trajectory $\mathbf{x}(\cdot)$ is enclosed in the corresponding tube $[\mathbf{x}](\cdot)$ if $\forall t \in [t] : \mathbf{x}(t) \in [\mathbf{x}](t)$.

Interval computations, as presented in Section 3.4, can easily be extended to tubes. In order to do so, we adapt the general definition for an operator $\diamond \in \{+, -, \cdot, /\}$ that is applied to two tubes $[\mathbf{x}](\cdot)$ and $[\mathbf{y}](\cdot)$:

$$[\mathbf{x}](\cdot) \diamond [\mathbf{y}](\cdot) = [\{\, \mathbf{x}(\cdot) \diamond \mathbf{y}(\cdot) \in \mathbb{R}^n \mid \mathbf{x}(\cdot) \in [\mathbf{x}](\cdot), \mathbf{y}(\cdot) \in [\mathbf{y}](\cdot) \,\}]. \tag{3.33}$$

In words, we compute the smallest tube $[\mathbf{x}](\cdot) \diamond [\mathbf{y}](\cdot)$ containing all possible combinations of arbitrarily picking a trajectory $\mathbf{x}(\cdot)$ out of tube $[\mathbf{x}](\cdot)$ and a a trajectory $\mathbf{y}(\cdot)$ out of tube $[\mathbf{y}](\cdot)$. Other operators defined for interval computations can be extended accordingly.

To find the range of possible values in a specific time interval, we recite the *interval evaluation* of a tube according to Bethencourt and Jaulin [77]:

$$[\mathbf{x}]([t]) = [\{\, \mathbf{x}(t) \mid \mathbf{x}(\cdot) \in [\mathbf{x}](\cdot),\, t \in [t] \,\}] = \bigsqcup_{t \in [t]} [\mathbf{x}](t). \tag{3.34}$$

Fig. 3.12 visualizes this operation. Another operation specifically designed for the handling of tubes is the *tube inversion*. Rohou [68] defines it as follows:

$$[\mathbf{x}]^{-1}([\mathbf{y}]) = \bigsqcup_{\mathbf{y} \in [\mathbf{y}]} \{\, t \in [t] \mid \mathbf{y} \in [\mathbf{x}](t) \,\}. \tag{3.35}$$

In words, we are interested in finding the time interval $[t]$ for which the tube intersects a given interval $[\mathbf{y}]$. Fig. 3.13 illustrates the idea. As can be seen, the resulting set is technically disconnected (e.g. there is a gap between $[t_1]$ and $[t_2]$), but since we aim to find a single interval enclosing all points in time, we have to settle for the interval enclosure of the single time intervals.

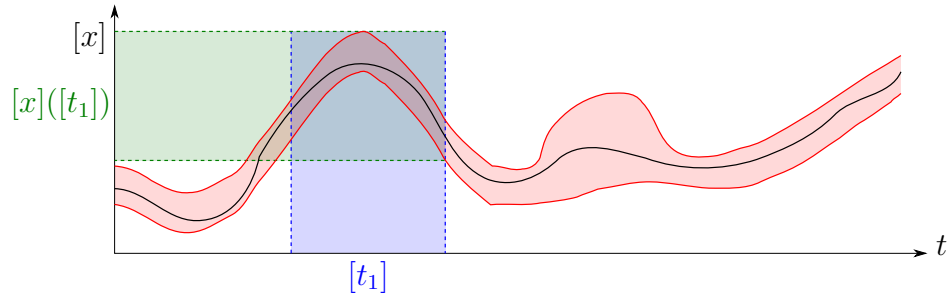


Figure 3.12: Interval evaluation of a tube for $[t_1]$.

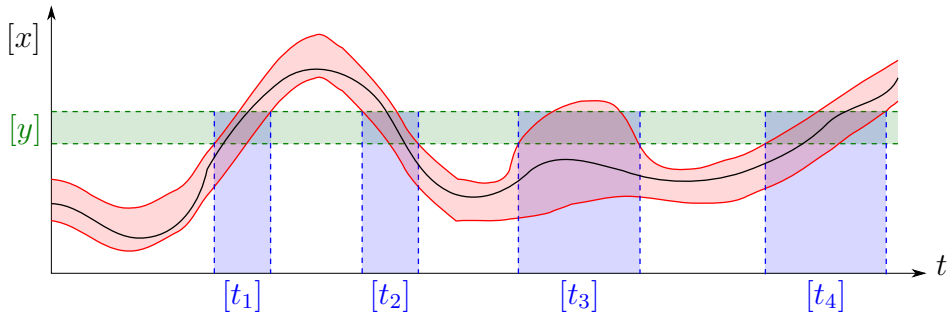


Figure 3.13: Tube inversion: $[x]^{-1}([y]) = \bigsqcup_{1 \leq i \leq 4} [t_i] = [\underline{t_1}, \overline{t_4}]$.

Generally, this operation is defined over the whole time interval $[t] = [t_0, t_f]$. However, in specific cases we may be interested in computing the tube inversion only for a predefined time interval $[t_p]$. In this case, it is defined as:

$$[\mathbf{x}]^{-1}([\mathbf{y}], [t_p]) = \bigsqcup_{\mathbf{y} \in [\mathbf{y}]} \{\, t \in [t_p] \mid \mathbf{y} \in [\mathbf{x}](t) \,\}. \tag{3.36}$$

It is evident that this definition is generally different to computing $([\mathbf{x}]^{-1}([\mathbf{y}])) \cap [t_p]$.

Similar as for static state estimation problems, it is possible to design *tube contractors* that can be applied to tubes. However, instead of contracting individual intervals, these contractors aim to remove parts of the tube, and thus remove whole trajectories from the set of trajectories. Generally, there are two different types of constraints for which contractors have been built. On the one hand, *synchronous* constraints are of the form $\mathbf{x}(t) = \mathbf{y}(t)$. On the other hand, *asynchronous* constraints are of the form $\mathbf{x}(t) = \mathbf{y}(t+1)$. The reader interested in the corresponding minimal contractors is referred to [77].

3.10.1 From real data to tubes

Contrary to tubes, which are defined continuously over time, real sensors output data at a fixed rate. Since we only know the measurement value (or interval) acquired at specific points in time, generally we cannot make a statement how the observed entity develops in between. This is especially true for sensors which provide data at a low frequency. Thus, the question arises how to fill the "gaps" between measurements to obtain a continuously defined function.

To answer this question, we have to first decide whether we are interested in finding a guaranteed enclosure that is suitable for a sound proof or whether we can settle for a fine

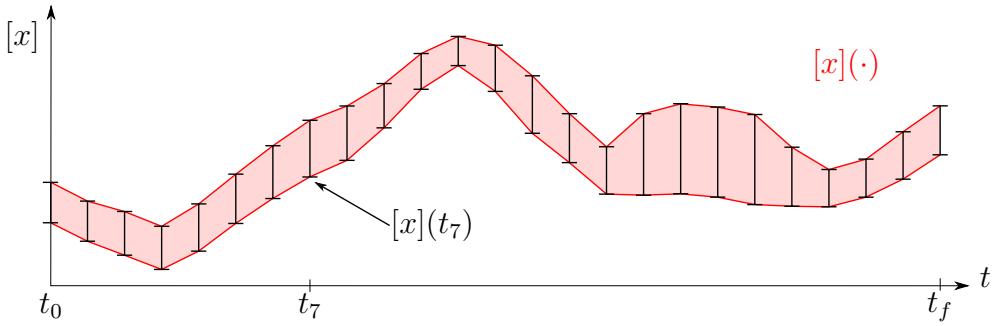


Figure 3.14: Linear interpolation between sensor measurements (black intervals) to create a tube that is continuously defined over time.

approximation that works well in practice. In the first case, derivative information is required to bound the progression of the observed value between measurements. Naturally, the derivative can be given in the form of a tube. For example, we may use GNSS measurements to obtain position estimates at a specified rate and simultaneously employ an odometer to additionally measure our robot's velocity. In this case, we are to build a tube that is *guaranteed* to contain the robot's true.

However, derivative information is often hard to obtain. In the case that it is not available, we have to settle for a fine approximation of the trajectory, which is usually sufficient - especially if the sensor outputs measurements at a high rate. Similar to Rohou [68], we linearly interpolate between the measurements to create a tube that is defined continuously over time. Fig. 3.14 illustrates the idea.

To represent tubes in a computer we use the *Tubex* library [78], which is openly available and compatible with IBEX. It implements a tube using slices of arbitrary widths as presented in [68, 77]. In other words, the tube is sampled into several interval boxes that have an arbitrary width on the time axis. This allows easier computation since interval arithmetic on boxes is well defined.

3.11 Importance of error bounds

Often, we do not have all the necessary information to accurately determine the error bounds of a measured value. For example, when measuring a distance using the ToF of a laser beam, the distance depends on the speed of light. However, the speed of light is influenced by many different factors (e.g. temperature, air pressure, etc.) that we may not know. Consequently, it becomes difficult to find accurate bounds for the speed of light and thus for the distance measured.

Since the most important assumption of interval analysis is that the bounds are guaranteed to enclose the true value, they should therefore be chosen conservatively, as to avoid a violation of the assumption. However, if the bounds are chosen too conservatively, the measurement is diluted, and even repeating the measurement does not increase its accuracy. In the best case, we know the exact error bounds that are reached during the measurement, but not exceeded. Then, the intersection of a several times repeated measurement becomes a degenerate interval and we know the exact true value. However, in practice the exact maximum error is almost never

known. Thus, the error bounds should be chosen conservatively, bearing in mind that a gross over-estimation is also undesirable. The following example demonstrates the aforementioned properties.

Example 3.11.1. Let $d_1 = 1.00\,\text{m}$, $d_2 = 1.01\,\text{m}$ and $d_3 = 0.99\,\text{m}$ be three measurements of the same distance with an actual accuracy of $\delta = 0.01$. However, we erroneously choose $\delta' = 0.1$ as our interval bounds. Consequently, the intersection of all three interval distances becomes

$$[d_1] \cap [d_2] \cap [d_3] = [0.9, 1.1]\,\text{m} \cap [0.91, 1.11]\,\text{m} \cap [0.89, 1.09]\,\text{m} = [0.91, 1.09]\,\text{m}. \quad (3.37)$$

As can be seen, the accuracy (i.e. interval width) of the intersected measurements is roughly the same as for the individual measurements (0.18 m vs 0.2 m). If we instead choose the correct accuracy of $\delta = 0.01$ we get

$$[d_1] \cap [d_2] \cap [d_3] = [0.99, 1.01]\,\text{m} \cap [1.00, 1.02]\,\text{m} \cap [0.98, 1.00]\,\text{m} = [1.00\,\text{m}]. \quad (3.38)$$

As can be seen, the interval becomes degenerate, thus resulting in the exact true value without any uncertainty.

3.12 Related work

In the following, we introduce state-of-the-art approaches for the spatiotemporal calibration of sensors and the dead reckoning of mobile robots using camera and/or laser scanner that employ interval analysis.

3.12.1 Spatiotemporal calibration

In 1985, Marzullo and Owicki introduced the first application of interval analysis for the synchronization of clocks in infrastructure networks [79]. They implement a service that allows to synchronize the time in client-server applications and consider the maximum error made during this synchronization. Although this work is not directly concerned with the time synchronization of black-box sensors and requires cooperation from the network nodes, we mention it here as it constitutes the first application of interval analysis in this field.

Bezet and Cherfaoui propose to timestamp sensor data using interval timestamps to take sensor latency, transmission delay and clock granularity into account [80]. Their goal is to convert the timestamps for sensor data from one time reference to another. Moreover, they continuously estimate the drift between sensor clocks. However, their approach requires a common clock, such as the bus network clock.

Olson presents an approach that does not use interval analysis tools but still conforms to the minimum/maximum error paradigm [35]. The author develops a passive algorithm that improves the computation of sensor data timestamps on the host computer. In contrast to other approaches, no cooperation from the sensor is required. Instead, the author's goal is to determine the communication latency which is the time that passes between the acquisition of sensor data and its arrival on the host computer. Given the assumption that each system

occasionally exhibits low-latency for a single sensor message, the minimum latency can be determined and constitutes a lower bound on the delay between sensor and host timestamps. Consequently, by considering this minimum latency, the sensor synchronization can only be improved. However, no upper bound for the synchronization error can be computed using this method.

Zaman and Illingworth propose the only approach that uses interval analysis and sensor data to find an interval for the relative timestamp offset between sensor clocks [81]. Their specific sensor combination in this work is an odometer and a camera. To detect the timestamp offset, the authors detect distinct events using both sensors. Consequently, they acquire timestamps in the time reference of each sensor that correspond to the same event. In their specific case this event is a change from non-motion to motion. Based on these event timestamps, they are able to compute an interval enclosing the timestamp offset. The accuracy of this interval depends on the measurement rate of each sensor, but can be improved by increasing the number of events. Later, it is proven that the algorithm manages to achieve a sub-sample accuracy [82]. Nevertheless, a large number of events is required to accurately estimate the timestamp, which requires extensive experiments on the one hand and may not be convenient to detect for other sensor combinations on the other hand.

So far, we have only presented related work on the time synchronization of sensors. This is because, to our best knowledge, no work has been published on the extrinsic calibration of sensors using interval analysis.

3.12.2 Visual-LiDAR odometry

Several interval-based approaches for the localization of a mobile robot have been presented. For example, Langerwisch and Wagner propose to use a laser scanner and wheel odometry to localize a robot in a two-dimensional feature-based map [83]. Similarly, Kenmogne et al. localize an unmanned aerial vehicle using a camera and known landmarks in the environment [84, 85]. However, in this work we focus on the dead reckoning of a mobile robot in a local coordinate system without a known map or landmarks.

Since the odometry computation is the first step of the SLAM problem, we also cite related work from this research field. In 2009, Jaulin proposed an approach to tackle the SLAM problem for underwater robots [86]. In this work, he casts the problem into a CSP and builds contractors that are able to reliable compute a box enclosing the robot's position. To compute the robot's odometry, a loch-Doppler and a gyroscope that measure the speed and rotation, respectively, are employed. In 2015, Jaulin proposed another approach to solve the SLAM problem [87]. In this work, he relies solely on range measurements and uses the robot's control inputs to compute an initial estimate of the robot's odometry. Subsequently, this estimate is contracted using information provided by the range sensors and from the built map.

Similar to Jaulin, Vincke et al. cast the SLAM problem into a CSP [88]. However, they use different sensors, namely two odometers and a camera. The odometers are employed to estimate the robot's odometry and the camera is used to build a map of distinct image features which are then used to correct the odometry estimate using forward-backward contractors.

Bethencourt and Jaulin introduce a set-membership approach for the three-dimensional reconstruction of arbitrary objects using a Microsoft Kinect, which is a device composed of a

color camera and an infrared depth sensor [89]. To be able to correctly reconstruct an object, it must be detected from different angles, thus requiring the motion between these viewing angle to be known. To estimate this motion, the authors detect visual features, for which they automatically receive depth information thanks to the infrared sensor, and match these across multiple image frames. Consequently, the motion can be estimated by computing the rigid body transformation. For this computation, the authors employ a forward-backward contractor.

Similarly, Mustafa et al. propose an approach to compute a robot's two-dimensional rigid body transformation between distinct points in time, which allows them to infer the robot's motion [90]. For their work, they assume to be able to detect distinct landmarks from different positions using range imaging sensors (infrared sensors, structured-light sensors, etc.). Subsequently, these associations allow them to formulate a CSP which is then solved using a forward-backward contractor. Later, the authors extend their work to the SLAM problem and prove that their approach for mapping is guaranteed to converge if some landmarks with known locations are available [91].

3.12.3 Relation to this work

In Section 3.12.1 we introduce approaches for the time synchronization of sensors using interval analysis. However, there is only one approach that computes an interval for the timestamp offset using only sensor data without requiring cooperation from the sensors. Moreover, this approach requires the detection of distinct events that need to be generated manually in extensive experiments. Contrary, this work proposes a new approach to compute the timestamp offset between camera and IMU, which does not require discrete events but relies on a continuous representation, namely the rotation of the sensors. In addition, we extend this approach to a spatiotemporal calibration by also computing the extrinsic rotation between sensor coordinate systems since, to our best knowledge, there exists no interval-based approach for the extrinsic calibration of sensors. For this reason, we also present a new approach for the extrinsic calibration of camera and laser scanner.

In Section 3.12.2 we present approaches to compute the odometry of a robot. Since the dead reckoning is a first step to solve the SLAM problem, most presented approaches come from this research field. Although different sensors are employed for every approach, they do not fuse information from these sensors on the measurement level. Instead, they compute an interval box enclosing the robot's pose using one sensor, and subsequently contract this box using information from a different sensor. In contrast, we propose to directly fuse information from camera and laser scanner due to the advantages outlined in Section 1.2. In this regard, the approach proposed by Bethencourt et al. and Mustafa et al. is closest to ours. Similar to us, they compute the rigid body transformation of a robot using corresponding 2D/3D points. However, they do not need to perform manual sensor fusion since they use range imaging sensors that directly provide color and distance information in an image. But, due to their short range and other drawbacks detailed in Section 1.2, these sensors are not suitable for use on autonomous vehicles.

4 Sensor Error Models

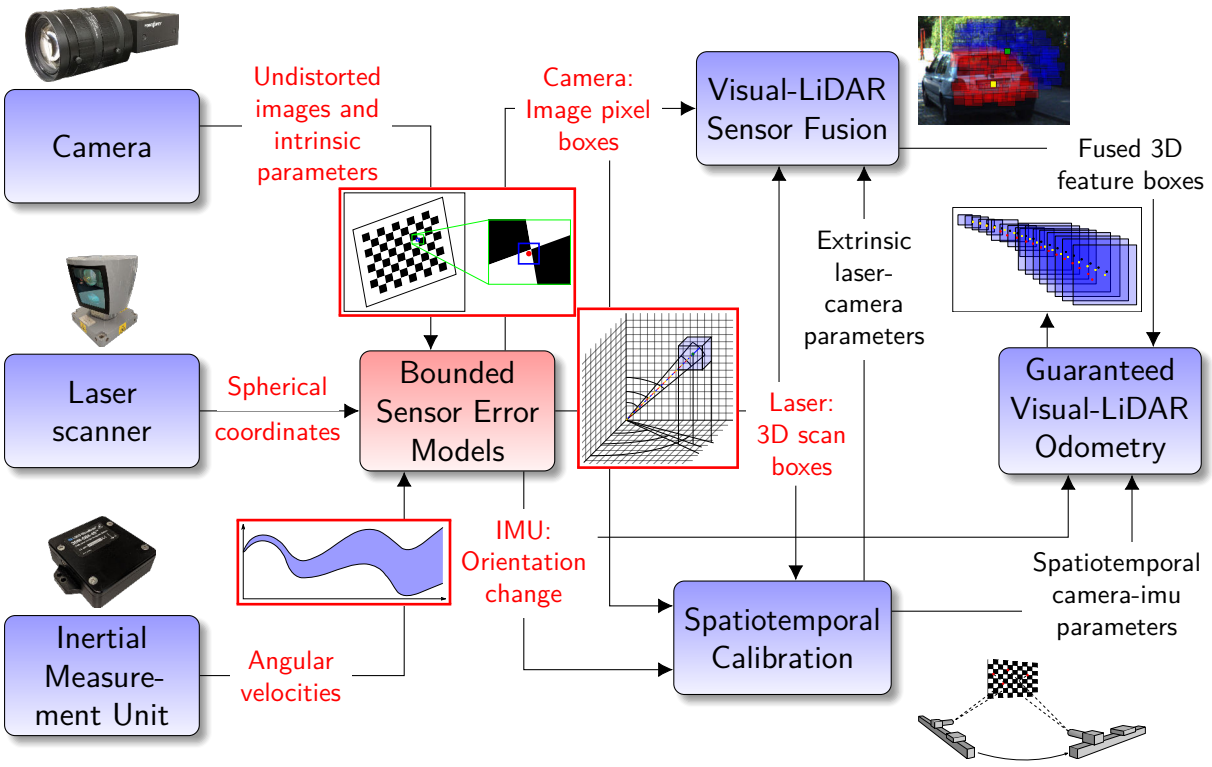


Figure 4.1: Classification of the sensor error models in the overall context of this work.

Since this work aims to develop methods for robust and reliable sensor fusion, sensor errors must be precisely modeled. While doing this, it is important to consider all expected sensor errors to find guaranteed bounds to uphold in the assumed conditions. Only if this prerequisite is met, interval analysis allows us to compute guaranteed results by consistently propagating errors from input to output sources.

This chapter summarizes the types of error that we consider for the sensors introduced in Section 2.1 and analyzes whether a stochastic or bounded error model is better suited to model this type of error. Subsequently, the sensor error models employed in this work are presented.

4.1 Types of error

Before guaranteed error bounds can be assigned to the output of each sensor, the expected error types for each observation have to be analyzed. Moreover, we compare the stochastic error model, which is predominant in robotics, to the bounded error model, which we employ for this work, in terms of applicability to each type of error.

4.1.1 3D laser scanner

Most types of error for a laser scanner can be quantified, i.e. we can determine a value for these errors and model them accordingly. However, outliers can occur due to non-reflective objects, mirrors or so called mixed pixels (also discontinuous points). These outliers cannot be described quantitatively, which is why Skrzypczynski [92] distinguishes between quantitative and qualitative error types for the laser scanner. In this work, we further subdivide the quantitative errors into systematic and random errors.

First, we will analyze the systematic errors and start with the offset between measurement and true value. The measurement of a 3D laser scanner is composed of the actual distance measurement and the inferred exit angle of the laser beam. For the distance measurement an offset can occur due to the incidence angle between the laser beam and the surface [16], and the environment of the sensor (temperature, humidity, etc.) [93]. Since this offset cannot be predicted without additional information about the environment, it cannot be modeled precisely. However, experiments and information from the manufacturer allow us to find error bounds enclosing the maximum offset. Thus, an interval based error model is consistent with this type of error. In contrast, a stochastic error model is inappropriate since an offset is a systematic error with an unknown error distribution (cf. Section 1.1.1 for a detailed explanation on the problem of modeling a systematic error using a stochastic distribution). Due to imperfect calibration parameters of the laser scanner, an offset can also occur for the exit angle of the laser beam. However, we assume the interior orientation parameters to be accurately determined [94], and thus consider the exit angle to be correctly reported by the laser scanner.

The second systematic error we consider is the beam divergence of the laser scanner. In theory, the laser scanner emits a perfect beam that has neither an initial footprint nor diverges. In reality, however, it is impossible to manufacture a sensor that emits a perfect beam, and so the laser beam hits not just a point but a surface. Since the power distribution inside the pulse is generally unknown, the actual location of the measured point is uncertain and could be anywhere within the beam footprint [95, 96]. Since the laser scanner only returns the point along the emitted beam centerline, the error distribution cannot be determined, and thus a stochastic error model is inappropriate. In contrast, an interval-based error model is suitable to model this type of error since the initial footprint and beam divergence are bounded and often specified by the manufacturer.

The only random error we consider for the laser scanner is the measurement noise [16]. Generally, this noise is assumed to follow a Gaussian distribution, however, this assumption needs to be verified by the sensor manufacturer or through experiments. Consequently, if the correct distribution of the measurement noise is determined, it can be modeled adequately using a stochastic error model. Nevertheless, an interval-based error model is also suitable

- although inferior in the case of a known error distribution - since the noise is generally not indefinite and appropriate error bounds can be derived from experiments or data sheets. Without reliable information about the error distribution, however, the bounded-error model is preferable because, unlike a stochastic model, it does not make false assumptions about the distribution. For example, this is the case for the Velodyne VLP-16, which we use for our experiments [96].

Finally, we compare both error models for qualitative errors. As defined above, qualitative errors are outliers that occur due to unpredictable situations in the environment. For example, mixed pixels occur when the laser beam hits two objects at different distances simultaneously. In this case, the laser scanner returns a distance that lies between both objects. Moreover, mirrors and non-reflective objects are undetectable by the laser scanner since they only reflect the laser beam in one direction or not at all. Since these outliers happen sporadically with a low probability, a stochastic distribution that incorporates a non-zero probability of any possible value (e.g. a Gaussian distribution) can be used to model them. However, the true probability of outlier occurrences is unknown, and thus a stochastic distribution can only serve as an approximation. The interval-based error model, however, is incompatible with outliers since it aims to provide guaranteed bounds for the measurement. Nevertheless, there exist interval analysis techniques to account for outliers if multiple measurements are available and the maximum number of outliers can be specified (cf. Section 3.9).

	Systematic		Random	Qualitative
Error model	Offset	Beam divergence	Measurement noise	Outliers
Stochastic	$-$	$-$	◯	◯
Interval-based	$+$	$+$	◯	$-$

Table 4.1: Comparison of the stochastic and the interval-based error model in terms of applicability to the different types of laser scanner errors. "$+$" signals a good, "◯" a reasonable and "$-$" a bad compatibility of the respective error model with the respective type of error. This table is inspired by Langerwisch [97].

Table 4.1 summarizes the findings of this section - namely the applicability of the stochastic and interval-based error model to the different types of laser scanner errors.

4.1.2 Camera

This work employs camera images to find features for both the spatiotemporal calibration and the visual-LiDAR sensor fusion. During the process of acquiring those image features, we consider different types of errors that can be classified into three categories. First, systematic errors arising from uncertain intrinsic parameters, quantization into image pixels and blur are analyzed. Second, both error models are compared for the image noise which constitutes a random error. Third, we examine both error models for applicability to outliers that occur during feature extraction or feature matching. Although many additional types of error could be analyzed in this section, we focus on those that we have identified as most important.

Besides, it is assumed that lens distortion is accurately estimated during camera calibration and does not vary during the experiment (cf. Section 2.1.2). Thus, the distortion parameters are not considered as an additional type of error.

As introduced, we distinguish between three different types of systematic errors. First, the camera intrinsic parameters can be uncertain - i.e. errors can occur during the camera calibration process or the intrinsic parameters can deviate due to mechanical stress. As explained in Section 2.1.2, the intrinsic camera parameters (focal lengths and principal point) are required to find the mapping between pixel coordinates in the image and camera coordinates. Consequently, since the camera calibration is performed once before carrying out the actual experiment and the calibrated intrinsic parameters are usually kept fixed throughout the experiment, an error in these intrinsic parameters is purely systematic [98]. Thus, the error cannot be described using a distribution and the stochastic model is inapplicable to model this type of error. In contrast, the interval-based model is consistent with systematic errors and is therefore suitable for modeling inaccurate intrinsic parameters (cf. Section 1.1.1).

Another systematic error hampering the extraction and matching of image features is the quantization during imaging. Since the camera has a limited resolution, the analog signal (i.e. the actual scene) has to be discretized into single pixels. Here, the analysis of the applicability of both error models to this type of error is straightforward. Since the quantization provides natural bounds enclosing the feature, the interval-based error model is applicable. Likewise, a stochastic error model is usable since the probability distribution of a feature inside a discretized pixel is known to be a uniform distribution [99].

The third systematic error source we consider is blur. Images can be blurred due to motion or objects not being in proper focus [100]. Since the amount and the characteristics of the blur depend on many different factors, of which many cannot be modeled due to missing information from the environment, the correct probability distribution of the position of a feature cannot always be determined [101]. Furthermore, unrecognizable systematic errors can occur, e.g. when detecting features on moving objects in the scene. However, a stochastic error model can only assume zero-mean error with an arbitrary distribution, which is often chosen as the normal distribution with an experimentally verified standard deviation. Similarly, the bounds for the interval-based error model can only be determined experimentally. However, the assumption of knowing bounds enclosing the error that is made due to blur during feature extraction, is weaker than assuming the correct distribution to be known. Besides, these bounds are consistent with systematic errors. Thus, the interval-based model is better suited to account for the blur that influences the process of feature extraction and matching.

Next, the only random error source we consider for this work - namely the image noise - is analyzed. Image sensors (e.g. Charge-Coupled Device (CCD) image sensor or Complementary Metal-Oxide Semiconductor (CMOS) image sensor) are subject to many different noise sources which can corrupt the individual color channels of each image pixel, and thus hamper the extraction of image features. In principle, each noise source can be modeled with a different distribution to find a common stochastic distribution resembling the image noise [18]. Thus, a stochastic error model is appropriate to model this type of error. However, determining the distribution for every noise source may not always be possible. In this case, a suitable error distribution can only be approximated, which may lead to an underestimation of the error.

Similarly, according to Liu et al. bounds enclosing the maximum noise can be determined [102] making the interval-based model applicable. For unknown noise sources affecting the error distribution, the interval-based error may even be superior, as it makes no assumption about the error distribution other than that it is bounded. However, generally a normal distribution is sufficient to approximate the error made due to image noise reasonably well.

Finally, we compare both error models for outliers occurring during the feature extraction or feature matching. Although there exist many approaches to eliminate outliers before employing them for further computations, we have to analyze whether outliers are consistent with both error models since it cannot be guaranteed that all outliers will be identified. For example, during feature matching in an urban environment, specific image patches may appear more than once in the scene (e.g. fencing posts), causing the feature matcher to compute wrong associations. As explained earlier, although there exist techniques to account for outliers (cf. Section 3.9), the interval-based error model is incompatible with this type of error, and therefore inappropriate in this case. Similarly, the stochastic error model suffers from the fact that the probability of outliers is unknown. However, common stochastic distributions (e.g. the Gaussian distribution) incorporate a non-zero probability of outlier occurrences, which theoretically makes them compatible with this type of error.

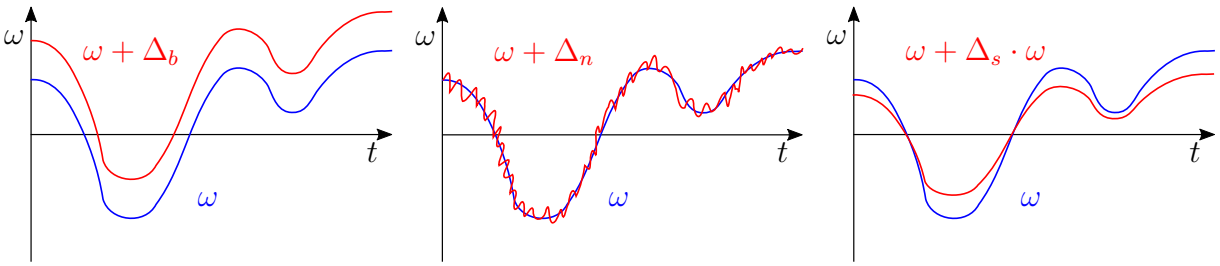
Table 4.2 summarizes the findings of this section - namely the applicability of the stochastic and interval-based error model to the different types of error occurring during the identification and matching of features in camera images.

	Systematic			Random	Qualitative
Error model	Intrinsic parameters	Quantization	Blur	Image noise	Outliers
Stochastic	−	+	−	◯	◯
Interval-based	+	+	◯	◯	−

Table 4.2: Comparison of the stochastic and the interval-based error model in terms of applicability to the different types of errors occurring when using features found in camera images. "+" signals a good, "◯" a reasonable and "−" a bad compatibility of the respective error model with the respective type of error.

4.1.3 Inertial Measurement Unit (IMU)

Since this work employs data only from the gyroscope of the IMU, we will only introduce the different types of error for this specific sensor. The literature usually distinguishes between systematic (deterministic) and random (stochastic) error factors. Deterministic errors occur due to physical processes such as imperfections during the manufacturing process. In contrast, stochastic errors depend on random fluctuations occurring only for a distinct period of time and can be related to environmental factors such as the temperature or humidity [103]. In general, estimating and removing the deterministic errors is most important for accurate IMU measurements, as remaining systematic errors cannot be modeled with a stochastic error model [104]. In contrast, the interval-based error model we propose is consistent with remaining systematic errors, as long as the bounds enclosing them are known (cf. Section 1.1.1).



(a) Bias/systematic error. (b) Noise (simplified; technically discontinuous). (c) Scale factor.

Figure 4.2: Examples of the different types of error exhibited by a gyroscope and their influence on the measured angular velocity ω.

Starting with the bias, we first focus on the systematic errors. A bias is an offset that is constant throughout all measurements of the sensor, but may change for each power-up of the IMU (turn-on to turn-on bias). Consequently, a simple calibration is necessary after each power-up: the IMU is kept static for several seconds, and thus the measurements should be zero. Any non-zero mean that is encountered during this calibration is labeled as bias and subtracted from all following measurements. However, due to the other error factors, the bias may be determined inaccurately which introduces an unknown systematic error. As unknown systematic errors are incompatible with any stochastic distribution (cf. Section 1.1.1), but are consistent with the interval-based error model, the bounded error model is better suited to model this type of error.

The second systematic error we consider is the scale factor, which describes the linear relation between true value (input) and sensor measurement (output). If no scale factor is considered, the output is equal to the input. In contrast, a scale factor of 2% means that the output is amplified by 2%. Similar to the bias, the scale factor can be determined during a calibration [105]. Consequently, any inaccuracies lead to a systematic error for which the interval-based error model is again preferable over any stochastic error model (cf. Section 1.1.1).

Next, we compare both error models for the random walk of the bias (also: bias instability). Due to various influence factors such as time, change in temperature or mechanical stress on the sensor, the bias, which we assumed above to be static, changes. Generally, the moving bias is characterized by Allan variance and modeled as a first order Gauss-Markov process [106], and is thus consistent with a stochastic error model. However, as many different factors - many of which cannot be observed - affect the bias stability, it is difficult to assess the true error distribution. In contrast, the interval-based error model does not require these influence factors to be known, but can still provide bounds reflecting the worst case of the bias drift [107]. Thus, depending on the known factors and the completeness of the stochastic error model, the more simple interval-based error model can be advantageous.

Measurement noise, which constitutes a random (stochastic) type of error, arises due to interferences of the internal electronics of the IMU with the actual signal. It is often assumed to be normally distributed [106], and thus a stochastic error model is adequate. Nevertheless, bounds enclosing the measurement noise can be determined to allow the use of the interval-based error model [107]. However, the stochastic error model is generally preferable for this type of error since the error distribution is well studied.

	Systematic		Random	
Error model	Scale factor	Bias	Walking bias	Noise
Stochastic	−	−	◯	+
Interval-based	+	+	◯	◯

Table 4.3: Comparison of the stochastic and the interval-based error model in terms of applicability to the different types of gyroscope errors. "+" signals a good, "◯" a reasonable and "−" a bad compatibility of the respective error model with the respective type of error.

Fig. 4.2 illustrates the different types of error. Subsequently, Table 4.3 summarizes the findings of this section - namely the applicability of the stochastic and interval-based error model to the systematic and random types of error.

4.2 Error models

After comparing the stochastic error model with the interval-based error model for the types of error we consider for each sensor, it becomes evident that the interval-based error model provides a reasonable alternative to the predominant stochastic error model. Especially in the case of unknown systematic errors the interval-based error model has an advantage over the stochastic error model. Therefore, this work develops and applies interval-based error models for each sensor. Consequently, interval analysis tools allow the consistent propagation of these sensor errors to the final results. However, the previous section also outlined problems of the interval-based error model - especially in the case of randomly distributed errors and outliers. For these types of error the stochastic error model is better suited. Thus, future work should deal with the question of how to couple the error models to benefit from the advantages of both.

In the following, this section introduces and explains the new sensor error models. As explained previously, some error sources (e.g. outliers) cannot be modeled correctly, and thus need to be considered in subsequent computations. While the 3D laser scanner error model is an extension of the 2D laser scanner error model introduced by Langerwisch [83], to our knowledge, the bounded-error models for camera and IMU (gyroscope) have not been presented to date.

4.2.1 3D laser scanner

As the error model for the laser scanner is an extension from the 2D laser scanner error model presented by Langerwisch, this section is inspired by his corresponding work [97].

As explained in Section 2.1.1, the raw measurements of a 3D laser scanner are the radial distance r, the polar (vertical) angle θ and the azimuthal (horizontal) angle φ. To obtain the 3D point corresponding to this measurement, these spherical coordinates can be transformed into Cartesian coordinates using (2.1).

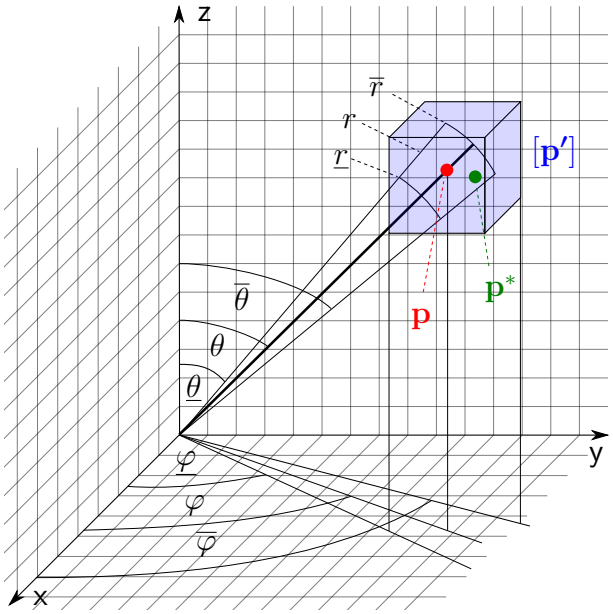


Figure 4.3: Visualization of the 3D box $[\mathbf{p}']$ that is guaranteed to enclose the actually measured point $\mathbf{p}^*$, which is different from the point $\mathbf{p}$ corresponding to the raw measurement. The initial footprint of the laser beam is not considered here.

As explained previously, the goal of this work is to apply intervals to model the uncertainties of the raw measurements. Amongst the error sources depicted in Section 4.1.1 are the systematic offset and the measurement noise which distort the distance measurement r. To model these two types of error, we determine a bounded error Δ_r that describes the maximum possible error such that the true distance r^* is enclosed in the interval $[r] = [r - \Delta_r, r + \Delta_r]$, $r^* \in [r]$. The last quantitative type of error we model using intervals is the beam divergence. It affects the polar and azimuthal angle since the true angle at which the returned distance was measured can be anywhere within the beam. Usually, the beam divergence is different in vertical and horizontal direction, and thus we define two different maximum possible errors Δ_θ and Δ_φ. Consequently, the true angles θ^* and φ^* are enclosed in the intervals $[\theta] = [\theta - \Delta_\theta, \theta + \Delta_\theta]$ and $[\varphi] = [\varphi - \Delta_\varphi, \varphi + \Delta_\varphi]$, i.e. $\theta^* \in [\theta]$ and $\varphi^* \in [\varphi]$.

Using $[r]$, $[\theta]$ and $[\varphi]$ it is now possible to compute a box $[\mathbf{p}']$ enclosing the true 3D point $\mathbf{p}^*$ assuming only the aforementioned types of error, but not yet the initial footprint of the laser beam. In order to do that, the real variables and operators in (2.1) are replaced by their interval counterpart:

$$[\mathbf{p}'] = \begin{pmatrix} [x'] \\ [y'] \\ [z'] \end{pmatrix} = \begin{pmatrix} [r] \cdot \sin[\theta] \cdot \cos[\varphi] \\ [r] \cdot \sin[\theta] \cdot \sin[\varphi] \\ [r] \cdot \cos[\theta] \end{pmatrix}. \tag{4.1}$$

Fig. 4.3 visualizes the computation of the 3D box $[\mathbf{p}']$ from uncertain spherical coordinates. Since each variable appears only once to compute $[x']$, $[y']$ and $[z']$, the intervals and thus the box are minimal. However, there exist dependencies between the three dimensions of the box. Generally, the geometric object enclosing the true point is not a box, but has a different form depending on the vertical and horizontal angle. Nevertheless, we compute the minimal box that encloses this arbitrary object, which simplifies further computations.

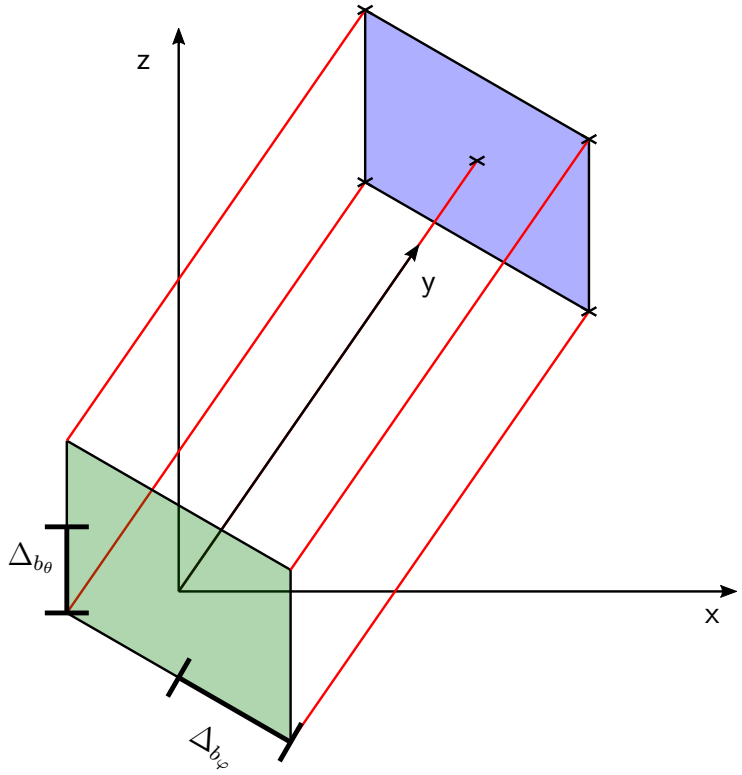


Figure 4.4: Exaggerated example for the influence of the laser beam footprint on the accuracy of the measured point. For simplicity, we assume no further divergence for the laser beam and no uncertainty for the distance measurement. Consequently, the vector pointing to the true measured point may not only originate from the origin of the laser scanner coordinate system, but from any point within the footprint.

To introduce the additional uncertainty arising due to the initial footprint of the laser beam, we find a rectangle that reflects this is initial footprint. Consequently, the vector pointing to the measured point may originate from any point within this rectangle, and just from the origin as assumed for (4.1). In order to find the rectangle, we compute two vectors that are orthogonal to the laser beam and point along the horizontal and vertical opening angle, respectively. This allows us to model different widths of the initial footprint in horizontal and vertical direction, which will be denoted as Δ_{b_φ} and Δ_{b_θ}. Fig. 4.4 shows an exaggerated rectangle corresponding to the footprint of the laser scanner and the resulting uncertainty of the measured point assuming no further uncertainties besides the initial footprint.

To find the two orthogonal vectors spanning the desired rectangle, we first denote $\mathbf{v}$ as the direction vector pointing to the measured point. $\mathbf{v}$ can be immediately deduced from (2.1):

$$\mathbf{v} = \begin{pmatrix} \sin\theta \cdot \cos\varphi \\ \sin\theta \cdot \sin\varphi \\ \cos\theta \end{pmatrix}. \tag{4.2}$$

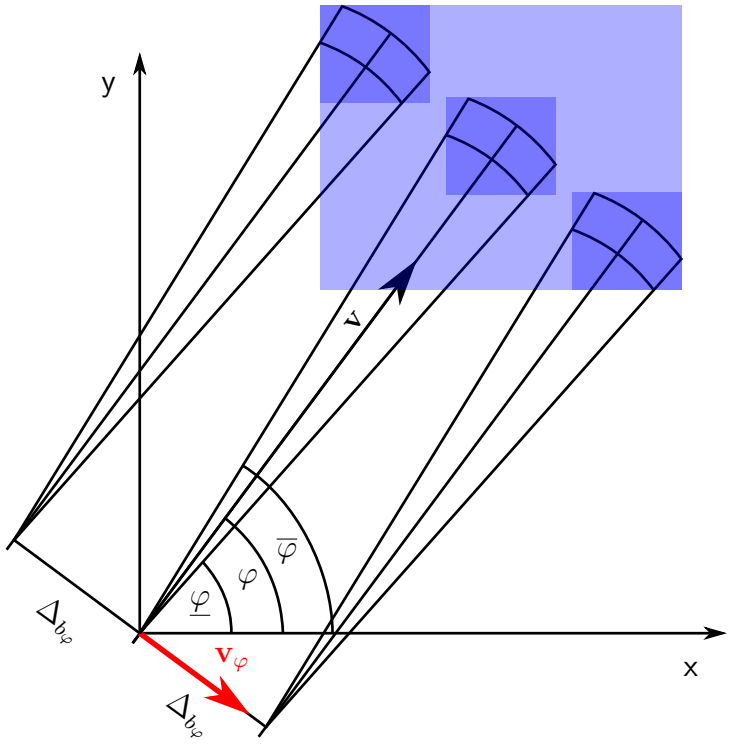


Figure 4.5: Exaggerated visualization of how the initial footprint of the laser scanner influences the uncertainty of the measured point. For simplicity, the figure shows a projection onto the x-y-plane and only highlights the horizontal expansion of the initial footprint. $\mathbf{v}_\varphi$ is the vector that is orthogonal to $\mathbf{v}$ and points along the horizontal axis of the laser scanner. Since the width of the footprint in horizontal direction is given as Δ_{b_φ}, we can compute the two extreme points from which the vector pointing to the measured point might have originated. Consequently, the size of the box containing the true point increases.

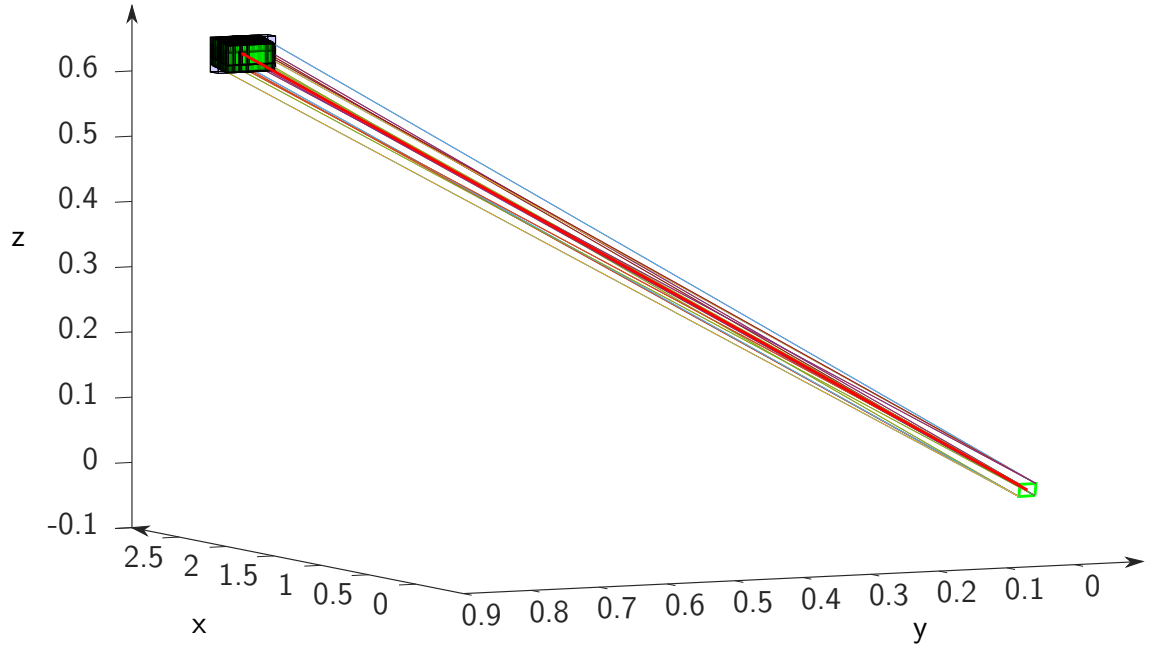
The first orthogonal vector, which we want to point along the horizontal opening angle, must reside in the x-y-plane. Fig. 4.5 shows a projection of $\mathbf{v}$ into the x-y-plane and the orthogonal vector $\mathbf{v}_\varphi$. Intuitively, it can be computed as

$$\mathbf{v}_\varphi = \begin{pmatrix} \sin\varphi \\ -\cos\varphi \\ 0 \end{pmatrix}. \tag{4.3}$$

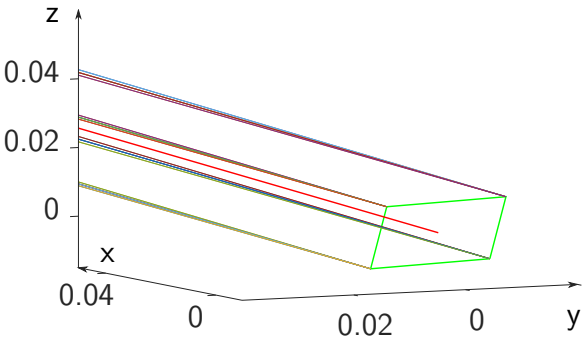
Subsequently, the second orthogonal vector $\mathbf{v}_\theta$, which points along the vertical opening angle, can be computed as the cross product between $\mathbf{v}$ and $\mathbf{v}_\varphi$:

$$\mathbf{v}_\theta = \mathbf{v} \times \mathbf{v}_\varphi = \begin{pmatrix} \cos\theta \cdot \cos\varphi \\ \cos\theta \cdot \sin\varphi \\ -\sin\theta \end{pmatrix}. \tag{4.4}$$

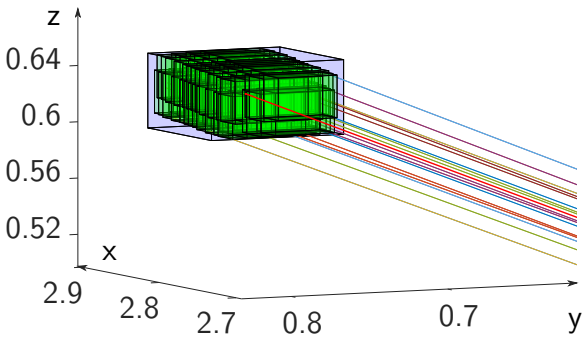
Finally, we can formulate the full equation for the laser scanner error model that allows to compute a box $[\mathbf{p}]$ enclosing the true measured point p^* given the measured distance r, vertical



(a) Overview.



(b) Detailed view of the initial footprint.



(c) Detailed view of the 3D box.

Figure 4.6: The most extreme laser beams originate from the most extreme points of the initial footprint of the laser scanner that is depicted as a green rectangle. The red laser beam corresponds to the measurement and starts at the origin. Subsequently, we bisect the intervals of all variables (distance, vertical and horizontal angle) to compute the green boxes, which approximate the shape of the geometric object describing the actual uncertainty of the measured point. The blue box corresponds to the computation in (4.5) and is the minimal 3D box enclosing all green boxes. As can be seen, some parts of the blue box correspond to an overestimation of the uncertainty that we must accept to simplify the following computations.

angle θ and horizontal angle φ, as well as the uncertainties for the distance measurement Δ_r, vertical angle Δ_θ, horizontal angle Δ_φ and the dimensions of the initial footprint Δ_{b_θ} and Δ_{b_φ}:

$$[\mathbf{p}] = \begin{pmatrix} [x] \\ [y] \\ [z] \end{pmatrix} = \begin{pmatrix} [r] \cdot \sin[\theta] \cdot \cos[\varphi] \\ [r] \cdot \sin[\theta] \cdot \sin[\varphi] \\ [r] \cdot \cos[\theta] \end{pmatrix} + [\Delta_{b_\varphi}] \cdot \mathbf{v}_\varphi + [\Delta_{b_\theta}] \cdot \mathbf{v}_\theta, \tag{4.5}$$

where $[\Delta_{b_\varphi}] = [-\Delta_{b_\varphi}, \Delta_{b_\varphi}]$ and $[\Delta_{b_\theta}] = [-\Delta_{b_\theta}, \Delta_{b_\theta}]$.

Since each variable appears only once as an interval to compute $[x]$, $[y]$ and $[z]$, the intervals and thus also the 3D boxes are again minimal. However, as explained previously, dependencies exist between the three components of the 3D box. Thus, we only compute the minimal box that encloses the geometric object, which is of an arbitrary shape.

We refrain from providing a proper visualization of the resulting computation in 3D since it is too complex to convey all information in one figure. However, the computation can be understood as a combination of Fig. 4.3, which shows the 3D box without considering the initial footprint, and Fig. 4.5, which considers all types of error, but shows the projection of the 3D box to the x-y-plane. Furthermore, Fig. 4.6 shows an example for the computation of the measurement box in 3D.

4.2.2 Camera

Since this work employs camera data only to find visual features, the raw measurements we use are pixel points (u, v) in the image. Consequently, the pinhole camera model (cf. Section 2.1.2) can be applied to obtain the homogeneous coordinates $(\,\tilde{x}\;\;\tilde{y}\;\;1\,)^\intercal$ in the camera coordinate system, which correspond to a 3D vector pointing in the direction of the image feature:

$$\tilde{x} = \frac{u - c_x}{f_x} \qquad\qquad \tilde{y} = \frac{v - c_y}{f_y}, \tag{4.6}$$

where (c_x, c_y) is the principal point and f_x, f_y are the focal lengths from camera calibration.

As introduced previously, we aim to formulate a bounded error model that allows to compute intervals which enclose the true homogeneous coordinates and further reflect their uncertainty. Thus, in the following we model all types of error introduced in Section 4.1.2. Previous work using bounded error models for camera data has been conducted by Telle et al. [98, 108, 109].

The first error source are uncertain intrinsic camera parameters which cannot be determined perfectly during camera calibration, and thus distort the transformation of the pixel points into homogeneous coordinates. Fig. 4.7 emphasizes this fact. Thus, we denote the maximum deviation of the focal lengths as Δ_{f_x} and Δ_{f_y}. Consequently, the intervals $[f_x] = [f_x - \Delta_{f_x}, f_x + \Delta_{f_x}]$ and $[f_y] = [f_y - \Delta_{f_y}, f_y + \Delta_{f_y}]$ enclose the true focal lengths ${f_x}^*$ and ${f_y}^*$. Similarly, the maximum deviations of the principal point are denoted as Δ_{c_x} and Δ_{c_y}. Therefore, the intervals $[c_x] = [c_x - \Delta_{c_x}, c_x + \Delta_{c_x}]$ and $[c_y] = [c_y - \Delta_{c_y}, c_y + \Delta_{c_y}]$ span the interval box $[c_x] \times [c_y]$ which encloses the true principal point $({c_x}^*, {c_y}^*)$.

The other types of error analyzed in Section 4.1.2 directly influence the detection of the image feature (u, v). Thus, we can model these uncertainties by determining the maximum deviations between the detected image feature (u, v) and the true location of the feature in the image

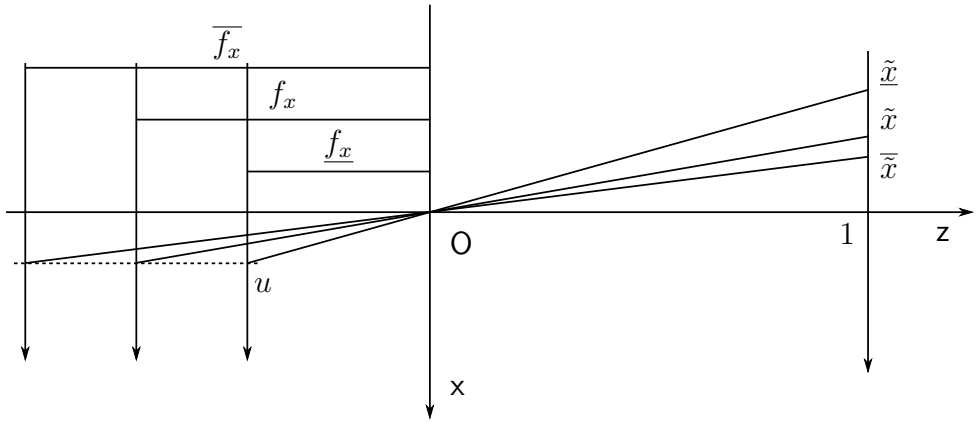


Figure 4.7: Geometry according to the pinhole camera model as seen from the y axis. The influence of an uncertain focal length on the homogeneous coordinates can be observed.

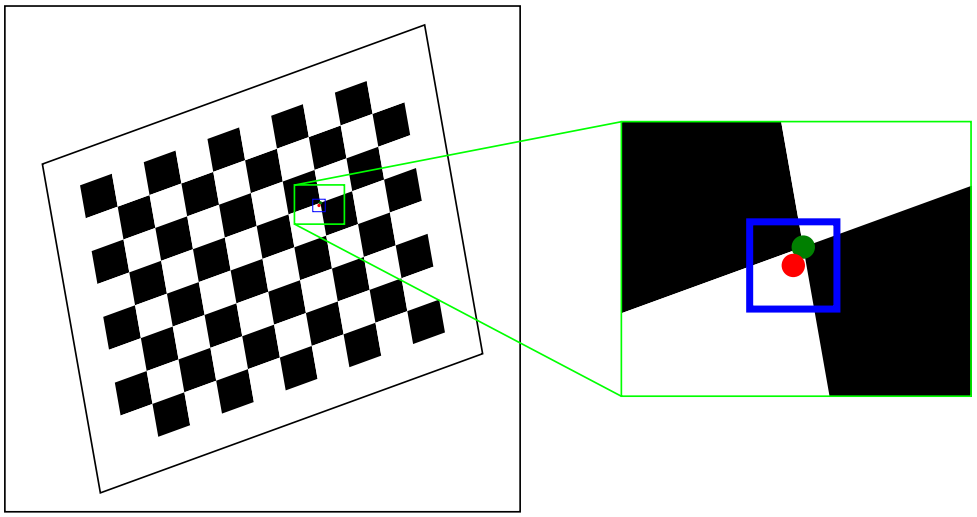


Figure 4.8: Exemplary image of a checkerboard for which the corner features (green dot) have to be detected. However, various error sources lead to an inaccurate detection (red dot). Thus, we apply our bounded-error model which results in the blue interval box that encloses the true checkerboard feature.

(u^*, v^*). These maximum deviations are denoted as Δ_{px} and are the sum of several individual error sources. Consequently, the interval box $[u] \times [v]$ encloses the true position (u^*, v^*) of the feature in the image, where $[u] = [u - \Delta_{px}, u + \Delta_{px}]$ and $[v] = [v - \Delta_{px}, v + \Delta_{px}]$. Fig. 4.8 shows an image feature that is detected inaccurately, but enclosed in the corresponding interval box. Likewise, Fig. 4.9 shows image features that are matched inaccurately, but their corresponding interval box overlap for parts of the same object.

As stated above, Δ_{px} is the sum of the individual types of error that are listed in Section 4.1.2, i.e.

$$\Delta_{px} = \Delta_{px_q} + \Delta_{px_b} + \Delta_{px_n}, \tag{4.7}$$

where Δ_{px_q}, Δ_{px_b}, and Δ_{px_n} are upper bounds for the feature detection error caused by quantization, image blur and measurement noise, respectively.

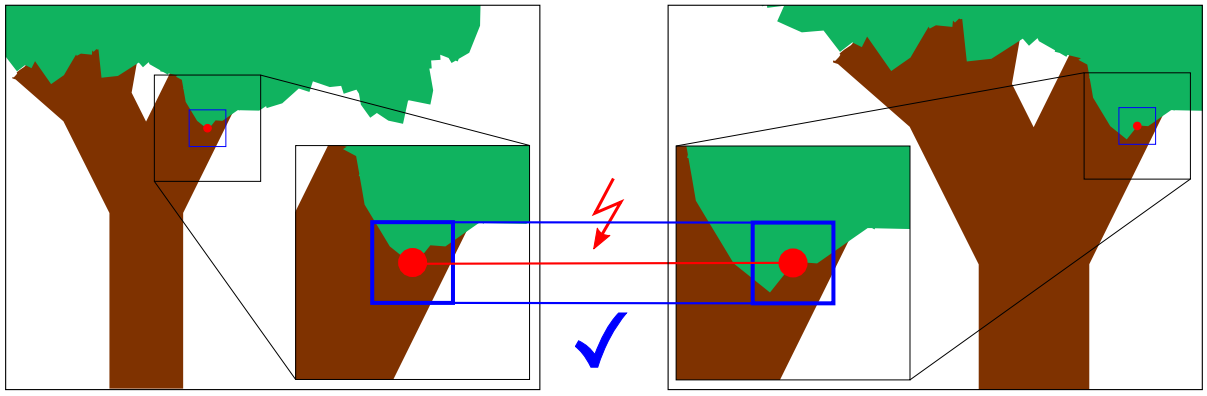


Figure 4.9: Two exemplary images for which matching image features are sought. The red dots are the anticipated result of traditional image feature matching and do not reside on exactly the same object in the real world due to the error sources mentioned previously. Thus, we use interval boxes (depicted in blue) to model the uncertainties. As can be seen, the two interval boxes overlap for parts of the same object, and therefore establish a correct feature matching.

Generally, $\Delta_{px_q} = 0.5\,\text{px}$ since in the worst case the actual location of the feature is on the bound of the pixel while the feature detector will report the midpoint of the pixel as the result. However, some features can be detected with sub-pixel accuracy by making use of known constraints (e.g. checkerboard corners reside on the intersection of lines). In this case, the quantization error Δ_{px_q} can be smaller than 0.5 px.

In contrast to the quantization error, the other two error sources affecting the feature extraction and matching - namely image blur and measurement noise - are more difficult to quantify. Therefore, we can only determine Δ_{px} empirically without distributing the error among the three individual types of error.

4.2.3 Inertial Measurement Unit (IMU)

As introduced in Section 4.1.3, we use only gyroscope data from the IMU which is distorted by two systematic errors, namely scale factor and bias, and two random errors, namely walking bias and noise. Thus, the only raw measurement we process is the 3×1 angular velocity vector $\boldsymbol{\omega}$ that contains the angular velocities around the three axis of the IMU. Consequently, we express the interval vector containing the true angular velocities as

$$[\boldsymbol{\omega}](t) = \boldsymbol{\omega}(t) + [\Delta_s] \cdot \boldsymbol{\omega}(t) + [\boldsymbol{\Delta}_b] + [\boldsymbol{\Delta}_{wb}] \cdot (t - t_0) + [\boldsymbol{\Delta}_n], \tag{4.8}$$

where t_0 is the start time, t is the time of the angular velocity measurement, $[\Delta_s]$ is the interval enclosing the scale factor, $[\boldsymbol{\Delta}_b]$ is the interval vector enclosing the constant bias, $[\boldsymbol{\Delta}_{wb}]$ is the interval vector enclosing the walking bias which can be time-dependent and $[\boldsymbol{\Delta}_n]$ is the interval vector enclosing the measurement noise.

Subsequently, these velocity measurements must be integrated to obtain the desired orientation of the IMU. However, as the gyroscope measures the angular velocities in a reference system that is local to the rotated IMU, the initial orientation of the IMU has to be considered. In order to do that, we use the interval extension of the first-order integration scheme detailed

in (2.12) in Section 2.1.3. As expected, the uncertainty of the computed rotation increases over time since the uncertainty of each velocity measurement is propagated to all successive points in time. This phenomenon is commonly known as *drift*.

4.3 Conclusion

This chapter provides original bounded sensor error models for three sensors commonly used in mobile robotics, namely the camera, laser scanner and IMU. These error models consider the different types of error we depict in the beginning of this chapter. Moreover, this chapter presents a comparison between an interval-based and a stochastic error model for the identified types of error. In summary, it can be stated that an interval-based error model is a viable model for each sensor, which is superior to a stochastic error model in the case of systematic errors.

In the context of this work, this chapter represents the part that is subject to failures, since the presented sensor error models form the backbone of all subsequent approaches. Consequently, these approaches are guaranteed to enclose the true solution only if the models are correct. In other words, the introduced error models constitute assumptions of the true sensor errors. Therefore, if these assumptions are violated, the following computations are no longer guaranteed. Ideally, in the future these sensor error models can be replaced by more sophisticated models which may come from a sensor manufacturer that is able to provide guaranteed error bounds. In this case, the following approaches for the spatiotemporal calibration of sensors and the visual-LiDAR odometry can adopt the more sophisticated models straightforwardly. The sole requirement is that the error model must supply bounds that are guaranteed to enclose the true value.

5 Perspective-n-Point (PnP) Problem

The idea of many spatiotemporal calibration approaches is to establish a connection between the data streams of both sensors by observing a known object, for example in a laboratory. If a camera is one of the sensors to be calibrated, this known object is often a checkerboard since it provides distinct features (the checkerboard corners) and it allows to compute the full 3D pose of the camera relative to the checkerboard by taking advantage of the fact that the dimensions of the checkerboard are known. Thus, the PnP problem (cf. Section 2.4) has to be solved before the actual calibration. Since this work introduces methods to perform a spatiotemporal calibration under interval uncertainty, the PnP problem must also be solved under interval uncertainty.

5.1 Constraints

To solve the PnP problem under interval uncertainty, we aim to build multiple contractors (cf. Section 3.7) that are based on different functions constraining the transformation between checkerboard coordinate system and camera coordinate system. As introduced in Section 2.4, this transformation consists of a 3×3 rotation matrix $\mathbf{R}_W^C \in \mathbf{SO}(3)$ (3 DOF, cf. Section 2.3) and a 3×1 translation vector $\mathbf{T}_W^C$ (3 DOF) and has a total of 6 DOF. In the following, this section introduces the constraints we identified and the corresponding contractors.

Let $\mathbf{X}^C = (\, x^C \;\; y^C \;\; z^C \,)^\intercal$ and $\mathbf{X}^W = (\, x^W \;\; y^W \;\; z^W \,)^\intercal$ be a pair of corresponding 3D points in the camera coordinate system C and the checkerboard/world coordinate system W. In total, we have n such corresponding 3D point pairs. The associated rigid body transformation can be expressed as:

$$\mathbf{X}^C = \mathbf{R}_W^C \cdot \mathbf{X}^W + \mathbf{T}_W^C. \tag{5.1}$$

However, as the camera only allows to measure the direction vector to a point, but not its 3D coordinates, we reformulate the equation using normalized coordinates $\tilde{\mathbf{X}}^C$ and an unknown scale factor λ:

$$\lambda \cdot \tilde{\mathbf{X}}^C = \mathbf{R}_W^C \cdot \mathbf{X}^W + \mathbf{T}_W^C, \tag{5.2}$$

where $\tilde{\mathbf{X}}^C = (\, \tilde{x}^C \;\; \tilde{y}^C \;\; 1 \,)^\intercal = (\, \frac{x^C}{z^C} \;\; \frac{y^C}{z^C} \;\; 1 \,)^\intercal$ are the normalized image coordinates as introduced in Section 2.1.2.

As $\tilde{\mathbf{X}}^C$ can be determined from the image by applying the pinhole camera model (cf. Section 2.1.2), it remains to remove the unknown scale factor λ. The third row of (5.2) can be expressed as:

$$\lambda = \mathbf{R}_3 \cdot \mathbf{X}^W + T_3, \tag{5.3}$$

where $\mathbf{R}_j$ and T_j are the jth row of $\mathbf{R}_W^C$ and $\mathbf{T}_W^C$, respectively.

Substituting (5.3) into (5.2) and reformulating allows us to find two equations without λ:

$$(\mathbf{R}_1 - \tilde{x}^C\mathbf{R}_3)\mathbf{X}^W + T_1 - \tilde{x}^C T_3 = 0, \tag{5.4}$$
$$(\mathbf{R}_2 - \tilde{y}^C\mathbf{R}_3)\mathbf{X}^W + T_2 - \tilde{y}^C T_3 = 0. \tag{5.5}$$

From here, there are two different possibilities to build contractors for these constraints. First, the rotation matrix can be expressed using a rotation representation introduced in Section 2.3 (e.g. Euler angles). This results in nonlinear equations for which we build forward-backward contractors. Second, stacking (5.4) and (5.5) for multiple corresponding image and world points results in a linear system for which we apply the Gauss-Seidel contractor [9].

5.1.1 Nonlinear forward-backward contractor

To reduce the number of unknowns for the rotation matrix, we express it using the Euler angles $\boldsymbol{\xi}_W^C = (\,\varphi_W^C \;\; \psi_W^C \;\; \theta_W^C\,)^\intercal$. Nevertheless, any other rotation representation introduced in Section 2.3 is possible. Subsequently, stacking the equations (5.4) and (5.5) for all n corresponding image and world points we obtain the constraints

$$\mathbf{f}_i^{\text{pnp,nl}}\left(\boldsymbol{\xi}_W^C, \mathbf{T}_W^C, \tilde{\mathbf{X}}_i^C, \mathbf{X}_i^W\right) = \mathbf{0}, \tag{5.6}$$

where $i \in \{1, \ldots, n\}$ and each $\mathbf{f}_i^{\text{pnp,nl}}$ has two rows resulting in a total of $2n$ constraints.

This allows us to formulate the CSP $\mathcal{H}_{\text{pnp,nl}}$:

$$\mathcal{H}_{\text{pnp,nl}} : \begin{pmatrix} \forall i \in \{1, \ldots, n\} : \quad \mathbf{f}_i^{\text{pnp,nl}}\left(\boldsymbol{\xi}_W^C, \mathbf{T}_W^C, \tilde{\mathbf{X}}_i^C, \mathbf{X}_i^W\right) = \mathbf{0} \\ \boldsymbol{\xi}_W^C \in [\boldsymbol{\xi}_W^C], \mathbf{T}_W^C \in [\mathbf{T}_W^C], \tilde{\mathbf{X}}_i^C \in [\tilde{\mathbf{X}}_i^C], \mathbf{X}_i^W \in [\mathbf{X}_i^W] \end{pmatrix}, \tag{5.7}$$

where $[\boldsymbol{\xi}_W^C]$ and $[\mathbf{T}_W^C]$ are the desired interval vectors enclosing the Euler angles and translation parameters, respectively. Furthermore, $[\tilde{\mathbf{X}}_i^C]$ are the known normalized camera coordinates of the observed point i. To obtain these, we use a state-of-the-art approach to detect checkerboard corners in the image, and subsequently apply the bounded error model derived in Section 4.2.2. Accordingly, $[\mathbf{X}_i^W]$ are the 3D coordinates in the checkerboard/world coordinate system of the same point i. Since the size of the squares of the checkerboard is known and the world coordinate system can be selected arbitrarily, these 3D coordinates can be computed straightforwardly. However, due to inaccuracies during the checkerboard manufacturing process, the 3D coordinates can exhibit an error that we model using intervals. Generally, the error bounds of these intervals correspond to the printing accuracy of the checkerboard.

Subsequently, we build a forward-backward contractor $\mathcal{C}_i^{\mathsf{pnp,nl}}([\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C])$ for each constraint i of the CSP $\mathcal{H}_{\mathsf{pnp,nl}}$ and intersect all of them to obtain the final contractor:

$$\mathcal{C}_{\mathsf{pnp,nl}}([\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]) = \bigcap_{1 \leq i \leq n} \mathcal{C}_i^{\mathsf{pnp,nl}}([\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]). \tag{5.8}$$

5.1.2 Linear Gauss-Seidel contractor

Another possibility to build a contractor for equations (5.4) and (5.5) is to take advantage of the fact that these are linear equations if we neglect the properties of the rotation matrix $\mathbf{R}_W^C$ and treat each of its entries as an unknown. This allows us to reformulate both equations as a system of linear interval equations:

$$\begin{pmatrix} x_1^W & y_1^W & z_1^W & 0 & 0 & 0 & -\tilde{x}_1^C x_1^W & -\tilde{x}_1^C y_1^W & -\tilde{x}_1^C z_1^W & 1 & 0 & -\tilde{x}_1^C \\ 0 & 0 & 0 & x_1^W & y_1^W & z_1^W & -\tilde{y}_1^C x_1^W & -\tilde{y}_1^C y_1^W & -\tilde{y}_1^C z_1^W & 0 & 1 & -\tilde{y}_1^C \\ \vdots & & & & & & \vdots & & & & & \vdots \\ x_n^W & y_n^W & z_n^W & 0 & 0 & 0 & -\tilde{x}_n^C x_n^W & -\tilde{x}_n^C y_n^W & -\tilde{x}_n^C z_n^W & 1 & 0 & -\tilde{x}_n^C \\ 0 & 0 & 0 & x_n^W & y_n^W & z_n^W & -\tilde{y}_n^C x_n^W & -\tilde{y}_n^C y_n^W & -\tilde{y}_n^C z_n^W & 0 & 1 & -\tilde{y}_n^C \end{pmatrix} \begin{pmatrix} R_{11} \\ R_{12} \\ R_{13} \\ R_{21} \\ R_{22} \\ R_{23} \\ R_{31} \\ R_{32} \\ R_{33} \\ T_1 \\ T_2 \\ T_3 \end{pmatrix} = \mathbf{0}, \tag{5.9}$$

where R_{ij} is the entry in the i-th row and j-th column of $\mathbf{R}_W^C$. As can be seen, each pair of corresponding image and world points introduces two rows in the matrix and thus the matrix is of dimension $2n \times 12$.

Unfortunately, finding the minimal solution for interval linear systems is NP-hard [110]. However, there exist efficient contractors to find a reasonably accurate solution. In this work, we apply the Gauss-Seidel contractor with preconditioning [9] which only works for square interval matrices on the left side of the equation (5.9). Thus, we need to reduce the number of rows of the matrix to twelve since the matrices contain twelve columns. In order to do that, we randomly choose six corresponding image and world point pairs since each pair introduces two rows in the matrix. If no preconditioning is used, the linear Gauss-Seidel contractor corresponds to the previously introduced forward-backward contractor.

Naturally, this procedure can be repeated many times with different point pairs until no further contraction of the entries of the rotation matrix R_{ij} and the translation vector $\mathbf{T}_W^C = (\, T_1 \;\; T_2 \;\; T_3 \,)^\intercal$ are achieved. We denote this contractor as $\mathcal{C}_{\mathsf{pnp,GS}}([\mathbf{R}_W^C], [\mathbf{T}_W^C])$. Although this contractor alone cannot provide a good enclosure for the transformation parameters since it ignores the properties of the rotation matrix, it can be used in conjunction with the nonlinear forward-backward contractor to provide an additional contraction.

5.2 Algorithm

After introducing the contractors in the previous section, we will now combine them in a common algorithm to solve the PnP problem under interval uncertainty. The algorithm employs both contractors and additionally bisects the Euler angles to improve the accuracy further. Thus, it is a version of the SIVIA algorithm introduced in Section 3.8. Algorithm 3 shows an overview of our algorithm.

Algorithm 3: PnP under interval uncertainty

input : $\mathcal{C}_{\text{pnp,nl}}$, $\mathcal{C}_{\text{pnp,GS}}$, $[\boldsymbol{\xi}_W^C]$, $[\mathbf{T}_W^C]$, ϵ
output : $\overline{\mathbb{X}}$, which is a subpaving for the 6 DOF transformation

1 $\mathcal{S} \coloneqq \left\{\left\{[\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]\right\}\right\}$; // $\mathcal{S}$ is a stack
2 **while** $\mathcal{S} \neq \varnothing$ **do**
3 $\quad \left\{[\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]\right\} \coloneqq \text{top}(\mathcal{S})$;
4 $\quad \left\{[\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]\right\} \coloneqq \mathcal{C}_{\text{pnp,nl}}\left([\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]\right)$;
5 $\quad [\mathbf{R}_W^C] \coloneqq \text{eul2mat}\left([\boldsymbol{\xi}_W^C]\right)$;
6 $\quad \left\{[\mathbf{R}_W^C], [\mathbf{T}_W^C]\right\} \coloneqq \mathcal{C}_{\text{pnp,GS}}\left([\mathbf{R}_W^C], [\mathbf{T}_W^C]\right)$;
7 $\quad \left\{[\boldsymbol{\xi}_W^C]\right\} \coloneqq \left\{[\boldsymbol{\xi}_W^C]\right\} \cap \text{mat2eul}\left([\mathbf{R}_W^C]\right)$;
8 $\quad$ **if** *any of* $\left\{[\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]\right\} = \varnothing$ **then**
9 $\quad\quad$ continue;
10 $\quad$ **else if** $w\left(\left\{[\boldsymbol{\xi}_W^C]\right\}\right) < \epsilon$ **then**
11 $\quad\quad \overline{\mathbb{X}} \coloneqq \overline{\mathbb{X}} \cup \left\{[\boldsymbol{\xi}_W^C], [\mathbf{T}_W^C]\right\}$;
12 $\quad$ **else**
$\quad\quad$ // only the Euler angles are bisected
13 $\quad\quad \left(\left\{[\boldsymbol{\xi}_{W,1}^C]\right\}, \left\{[\boldsymbol{\xi}_{W,2}^C]\right\}\right) \coloneqq \text{bisect}\left(\left\{[\boldsymbol{\xi}_W^C]\right\}\right)$;
14 $\quad\quad \mathcal{S} \coloneqq \mathcal{S} \cup \left\{\left\{[\boldsymbol{\xi}_{W,1}^C], [\mathbf{T}_W^C]\right\}\right\} \cup \left\{\left\{[\boldsymbol{\xi}_{W,2}^C], [\mathbf{T}_W^C]\right\}\right\}$;
15 $\quad$ **end**
16 **end**

At first, the user needs to specify initial intervals for the Euler angles and the translation vector, which are denoted as $[\boldsymbol{\xi}_W^C]$ and $[\mathbf{T}_W^C]$. Generally, these intervals can be arbitrarily large, but the Euler angles must be bounded in such a way that ambiguities cannot occur (cf. Section 2.3.1).

Subsequently, the standard SIVIA procedure is followed until we call the first contractor in Line 4. After employing the nonlinear contractor (cf. Section 5.1.1) to contract the domains for both the Euler angles and the translation vector, we have to convert the Euler angles to a rotation matrix before the linear Gauss-Seidel contractor is applicable. In order to do that, we make use of the interval extension (i.e. the natural inclusion function) of equation (2.16) that is depicted in Section 2.3.1. Consequently, the linear Gauss-Seidel contractor (cf. Section 5.1.2) is applied to further reduce the uncertainty of both the rotation matrix and the translation vector. Next, the rotation matrix must be converted back to Euler angles to keep the number of variables as low as possible and allow a more efficient bisection process. Equation (2.17)

shows the general conversion from a rotation matrix back to Euler angles. Thus, we use the interval extension of this function to compute the contracted Euler angles in Line 7.

Afterwards, our algorithm again follows the standard SIVIA procedure. However, instead of bisecting all six parameters (three Euler angles and three translations), we bisect only the Euler angles in Line 13. The reason is that each translation appears only once in equations (5.4) and (5.5), and thus can be computed more accurately using the nonlinear forward-backward contractor $\mathcal{C}_{\text{pnp,nl}}$. In fact, given an optimal enclosure of the Euler angles, the forward-backward contractor would suffice to compute the optimal domains for the translation parameters. In contrast, the Euler angles appear more often (each Euler angle appears multiple times in the rotation matrix, cf. equation (2.16) in Section 2.3.1), and are therefore subject to an overestimation which we aim to reduce by applying bisections.

5.3 Conclusion

This chapter provides an algorithm to contract the 6 DOF transformation between camera coordinate system and known 3D points in a world coordinate system. Since interval analysis tools, namely contractors and SIVIA, are employed, the computed intervals are guaranteed to enclose the true solution without requiring a good initial guess. This constitutes a benefit over existing stochastic approaches. Another contribution of this chapter is to randomly select points for the Gauss-Seidel contractor due to the problem being over-constrained. However, the combination of the forward-backward and the Gauss-Seidel contractors is not optimal since the decomposition of the problem into multiple independent constraints leads to an overestimation of the uncertainty. Consequently, SIVIA is required to compute a more accurate solution. In the future, it would be beneficial to introduce a special contractor dedicated to the PnP problem that is capable of computing an accurate solution without requiring bisections.

In the context of this work, the PnP problem must be solved to perform the spatiotemporal calibration between a camera and other sensors. In the following, we use the algorithm presented to compute the pose of a camera relative to a checkerboard that serves as a calibration target.

6 Spatiotemporal Calibration

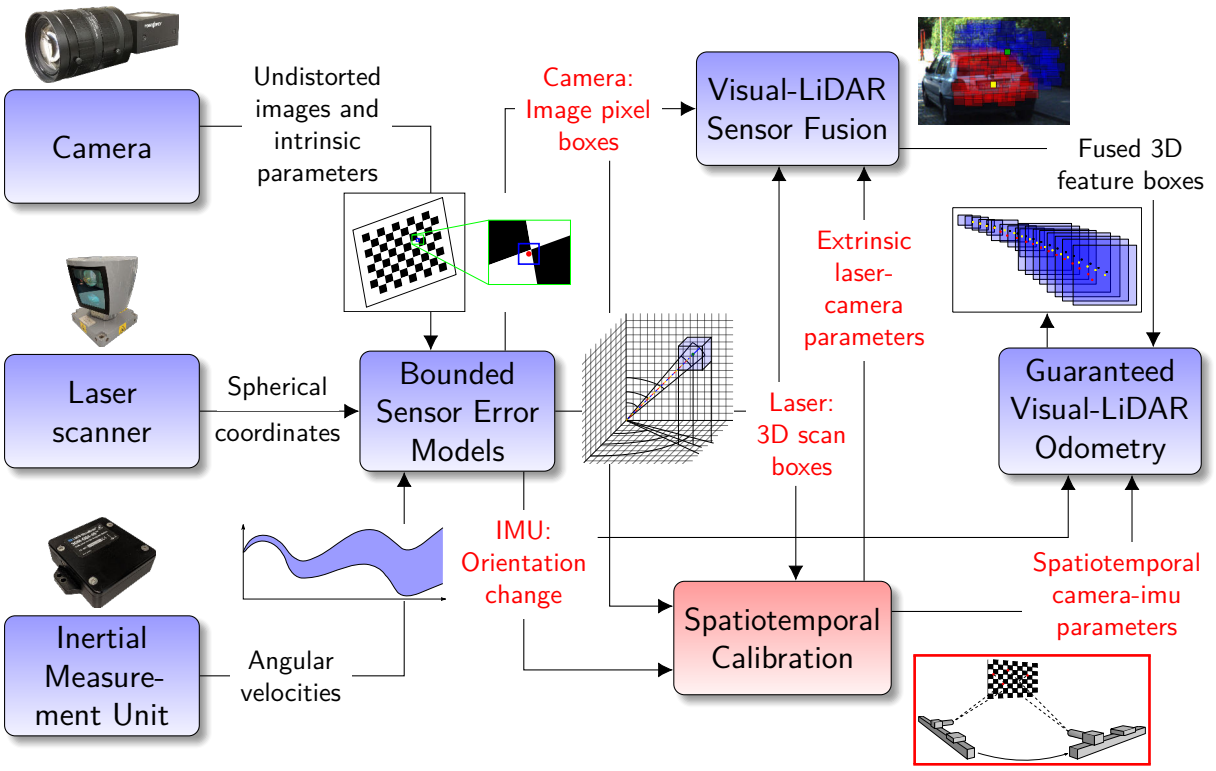


Figure 6.1: Classification of the spatiotemporal calibration in the overall context of this work.

The basis to perform sensor fusion is the knowledge of the spatiotemporal calibration parameters for the sensors from which information is to be fused. Generally, the parameters can be divided into two categories: spatial and temporal. As depicted in Section 2.5, spatial parameters usually concern the extrinsic transformation consisting of the rotation matrix $\mathbf{R}_B^A$ and the translation vector $\mathbf{T}_B^A$ between sensor coordinate systems A and B. Formalizing this relation in an equation yields

$$\mathbf{X}^A = \mathbf{R}_B^A \cdot \mathbf{X}^B + \mathbf{T}_B^A, \tag{6.1}$$

where $\mathbf{X}^A$ and $\mathbf{X}^B$ are 3D coordinates in the (sensor) coordinate systems A and B, respectively.

In contrast, temporal parameters describe the time relation between sensor clocks. As explained in Section 2.5, in this work we focus on the relative time offset τ since it is assumed to be constant over a short period of time. Furthermore, it can be re-estimated multiple times

to negate the effect of drifting sensor clocks. Thus, an event or an effect observed by sensor A at time t is registered by sensor B at time $t + \tau$:

$$x_A(t) = x_B(t + \tau). \tag{6.2}$$

As depicted in Section 1.1.2, we assume the sensors to be black-box systems and aim to rely solely on sensor data to determine the spatiotemporal calibration parameters. Accordingly, we presented several stochastic approaches to perform the extrinsic and/or temporal calibration between camera, laser scanner and IMU. However, since the goal of this work is to fuse information from camera, laser scanner and IMU under interval uncertainty to perform dead reckoning in a guaranteed way, the spatiotemporal parameters must be estimated under interval uncertainty as well. Thus, this chapter presents calibration procedures that are adapted to the bounded error models developed in Chapter 4.

First, Section 6.1 introduces our interval-based approach to compute the time offset and the extrinsic rotation parameters between camera and IMU. Determining the extrinsic translation parameters is not required as we will only use rotation measurements from the IMU to support the guaranteed localization. Subsequently, Section 6.2 presents a new approach to determine the extrinsic calibration parameters between camera and laser scanner under interval uncertainty. In this work, for the reasons given in Section 1.1.1, we assume camera and laser scanner to be synchronized (e.g. by triggering the image acquisition of the camera externally), and thus there exists no time offset that would have to be determined. Nevertheless, the procedure for determining the time offset between camera and IMU in Section 6.1 could straightforwardly be adapted to the data correspondences established between camera and laser scanner in Section 6.2.

6.1 Spatiotemporal calibration between camera and IMU

Since this work aims at fusing information from camera and IMU, the spatiotemporal calibration parameters between both sensors have to be known. Here, we consider a constant time offset τ as the temporal part of this spatiotemporal calibration. This temporal offset manifests itself in such a way that the same real world effect, which is observed by both sensors, is assigned the timestamp t by the camera and the timestamp $t + \tau$ by the IMU. Consequently, we have to deal with time uncertainties that have been identified to be generally difficult to deal with [111].

Given a one-dimensional effect that is observed by both sensors, it holds that

$$\forall t: \; x_C(t) = x_I(t + \tau), \tag{6.3}$$

where $x_c(\cdot)$ and $x_I(\cdot)$ are the effects observed by the camera and the IMU, respectively.

However, as this work employs angular velocities measured by the IMU to predict the rotation of the camera, the effect we are interested in is not one-dimensional, but constitutes a rotation in 3D. For example, the rotation of the IMU can be expressed as $\mathbf{R}_I^{I_0}(\cdot)$, where I_0 is the initial frame (relative to which we compute the rotation) and I is the current frame of the IMU. Since any rotation matrix can be represented using only three parameters (cf. Section 2.3), the effect for which we determine a constant time offset is three-dimensional.

Besides the temporal calibration, we need to simultaneously determine the rotation matrix $\mathbf{R}_C^I$ that describes the static rotation between the coordinate systems of both sensors. This information is required to later employ the rotation measured by the IMU in its coordinate system I to predict the rotation of the camera in its own coordinate system C. The rotation $\mathbf{R}_C^I$ is not time-dependent as the sensors are rigidly mounted, and thus the rotation between them does not change, but remains static. Since this work employs the IMU only to estimate the rotation, determining the translation between both coordinate systems is not required.

Summarizing the constraints for the spatiotemporal calibration in a common equation yields:

$$\forall t: \ \mathbf{R}_C^{C_0}(t) = \left(\mathbf{R}_C^I\right)^\intercal \cdot \mathbf{R}_I^{I_0}(t+\tau) \cdot \mathbf{R}_C^I, \tag{6.4}$$

where $\mathbf{R}_C^{C_0}(\cdot)$ and $\mathbf{R}_I^{I_0}(\cdot)$ are the rotations relative to the initial frames C_0 and I_0 over time of the camera and IMU, respectively.

In the following, Section 6.1.1 presents the idea to perform the spatiotemporal calibration between camera and IMU, which we previously published in [112]. Subsequently, Section 6.1.2 presents a new contractor $\mathcal{C}_{\text{offset}}$ to determine the constant time offset between two arbitrary effects for which an unknown but bounded error is assumed (i.e. tubes). Finally, Section 6.1.3 details how this contractor is employed in a version of the SIVIA algorithm to simultaneously estimate both the temporal offset and the spatial/extrinsic rotation between both sensors (previously published in [27]).

6.1.1 General idea

Since we assume the sensors to be black-box systems for which the relative time offset cannot be computed externally (e.g. using algorithms like the Network Time Protocol (NTP) [15]), we have to rely on sensor data that is timestamped either at a centralized receiver or by each sensor individually. Consequently, we aim to find correspondences in the data of both sensors, i.e. we require common data that can be measured by both camera and IMU and that allows us to deduce the constant time offset. Inspired by Kelly et al. [38], we use orientation measurements as a common representation in this work.

Fig. 6.2 shows the general idea. Both sensors are rigidly mounted on a common platform such that the extrinsic rotation $\mathbf{R}_C^I$ between their respective coordinate systems does not change. At the start of the experiment, t_0, this platform is kept static and we denote the corresponding, initial coordinate systems by C_0 and I_0 for the camera and IMU, respectively. Afterwards, the setup is rotated either manually or by using a motor.

Now, the idea is to compute the three-dimensional rotation over time for both sensors. In order to do that for the camera, a calibration target with known 3D world points is required. In this work, we choose a checkerboard as our calibration target and compute the camera pose relative to this target by solving the PnP problem under interval uncertainty (cf. Chapter 5) for each acquired camera image. This results in an estimate for the orientation of the camera relative to the checkerboard coordinate system over time: $\mathbf{R}_W^C(\cdot)$. Consequently, the rotation relative to the starting time t_0 can be expressed as

$$\mathbf{R}_C^{C_0}(\cdot) = \mathbf{R}_W^{C_0} \cdot \left(\mathbf{R}_W^C(\cdot)\right)^\intercal. \tag{6.5}$$

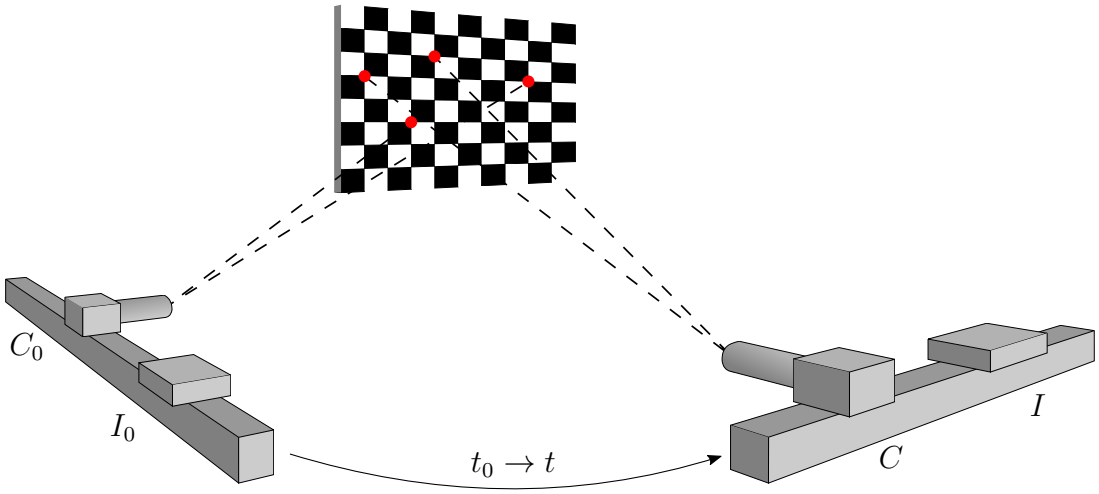


Figure 6.2: Visualization of the general idea to perform the spatiotemporal calibration between camera and IMU. Both sensors are mounted rigidly on a common platform, such that they experience the same movement during the experiment. Subsequently, we move the platform and compute the rotation of both sensors between the starting time t_0 and the current time t. For the camera, this rotation can be determined by observing a static checkerboard calibration target. For the IMU, the angular velocities are integrated. This results in three-dimensional tubes (orientation over time) for both sensors that constrain both the constant time offset and the constant extrinsic rotation parameters.

To compute the rotation over time for the IMU, we integrate the angular velocities under interval uncertainty as introduced in Section 4.2.3. In contrast to the camera, the uncertainty of the orientation estimates by the IMU increases over time due to the integration process.

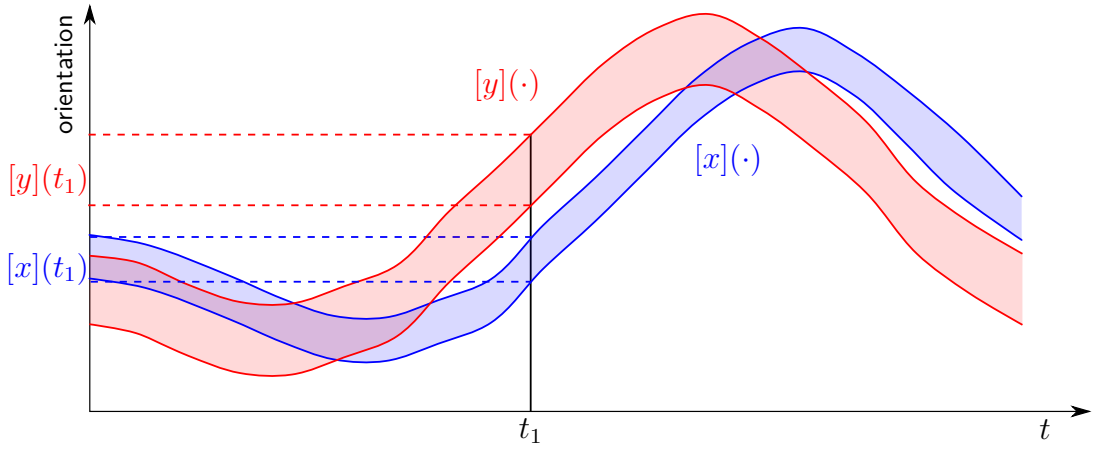
After acquiring three-dimensional tubes that bound the rotation over time for both sensors, the these tubes must satisfy the following constraint:

$$\begin{aligned} \forall t: \ \mathbf{R}_C^{C_0}(t) &= \left(\mathbf{R}_C^I\right)^\intercal \cdot \mathbf{R}_I^{I_0}(t+\tau) \cdot \mathbf{R}_C^I, \\ \mathbf{R}_C^{C_0}(\cdot) \in [\mathbf{R}_C^{C_0}](\cdot), \mathbf{R}_I^{I_0}(\cdot) &\in [\mathbf{R}_I^{I_0}](\cdot), \mathbf{R}_C^I \in [\mathbf{R}_C^I], \tau \in [\tau]. \end{aligned} \tag{6.6}$$

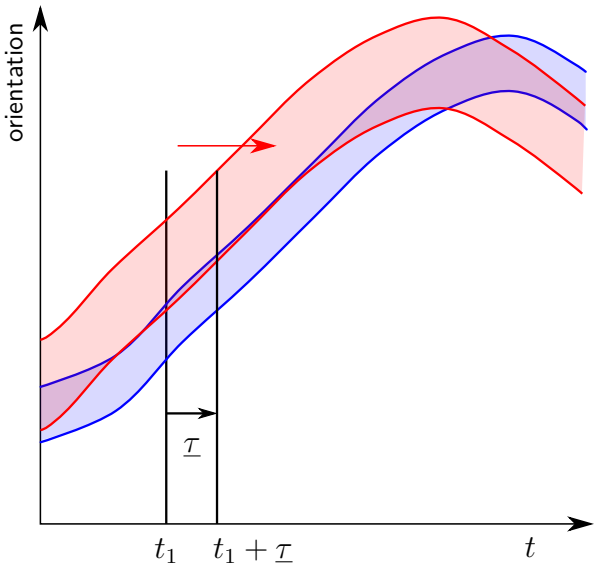
In simple terms, this means that the orientation tubes that are measured by the IMU and transformed into the camera coordinate system, must overlap the orientation tubes measured by the camera. However, as there is an unknown offset τ, these tubes are offset to each other on the time axis. Fig. 6.3a shows a one-dimensional example that ignores the unknown rotation $\mathbf{R}_C^I$ and focuses on the time offset. As can be seen, the camera and IMU tubes do not overlap, and thus there exists a non-zero time offset.

The general idea to find an interval enclosing the time offset $[\tau]$ is to shift one tube (e.g. the red one) along the time axis until it starts overlapping the other tube for every point in time. This allows to find the lower/upper (depending on the sign of τ) bound of $[\tau]$. Subsequently, we find the other (upper/lower) bound of $[\tau]$ by shifting the tube further in the same direction until it barely overlaps the other tube. Fig. 6.3b and Fig. 6.3c visualize this idea.

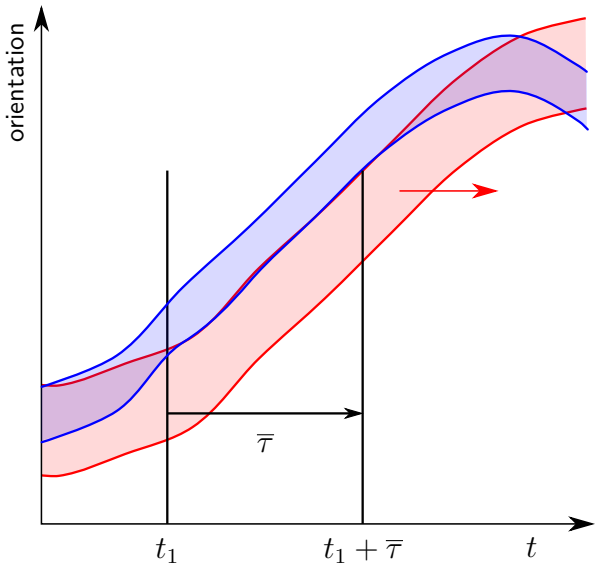
Generally, our algorithm does not require tubes from orientation measurements, but works with any common data that can be measured by both sensors. Moreover, the orientation tubes



(a) Exemplary visualization of two one-dimensional tubes that exhibit a time offset. $[x](t_1)$ and $[y](t_1)$ are the interval evaluations of the tubes $[x](\cdot)$ and $[y](\cdot)$ at time t_1, respectively. Since these intervals do not overlap, it is evident that there exists a non-zero time offset between both tubes.



(b) To find the lower bound of $[\tau] = [\underline{\tau}, \overline{\tau}]$, the red tube is shifted along the time axis (to the right) until it overlaps the blue tube for every point in time.



(c) To find the upper bound $\overline{\tau}$, the red tube is further shifted along the time axis (to the right) until it barely overlaps the blue tube for every point in time.

Figure 6.3: Overview over the idea to determine the time offset between two one-dimensional tubes.

could be determined using natural image features, thus removing the need for a calibration target and enabling a periodic re-estimation of the timestamp offset on a moving vehicle to negate a possible clock drift.

In the following, Section 6.1.2 introduces a new contractor $\mathcal{C}_{\text{offset}}$ that computes the desired time offset domain $[\tau]$ between two one-dimensional tubes. Consequently, in Section 6.1.3 this contractor is coupled with SIVIA to compute both the unknown rotation between sensor coordinate systems $[\mathbf{R}_C^I]$ and the time offset $[\tau]$ from three-dimensional orientation tubes.

6.1.2 Time offset contractor

This section introduces a new contractor for the CSP $\mathcal{H}_{\text{offset}}$ that is defined as:

$$\mathcal{H}_{\text{offset}} : \begin{cases} \textbf{Variables: } \tau,\, x(\cdot),\, y(\cdot) \\ \textbf{Constraints:} \\ \quad 1.\ \ \forall t \in [t_0, t_f] : x(t) = y(t+\tau) \\ \textbf{Domains: } [\tau],\, [x](\cdot),\, [y](\cdot) \end{cases}, \tag{6.7}$$

where $[t_0, t_f]$ is the time interval over which both $[x](\cdot)$ and $[y](\cdot)$ are defined. In theory, this CSP introduces an infinite number of constraints due to the $\forall$-operator. However, in practice we sample the time domain $[t_0, t_f]$, and thus obtain a finite number of constraints.

In words, $x(\cdot)$ and $y(\cdot)$ are two trajectories (or effects) that are shifted in time by a constant offset τ. Although the trajectories are one dimensional in this definition, the CSP (and thus the corresponding contractor) can be extended to the multidimensional case straightforwardly. In the following, we present a contractor $\mathcal{C}_{\text{offset}}$ that can be employed to contract both the time offset domain $[\tau]$ and the tubes $[x](\cdot)$ and $[y](\cdot)$ without dismissing part of the solution.

Proposition 1

The operator $\mathcal{C}_{\text{offset}}$ is a contractor for the CSP $\mathcal{H}_{\text{offset}}$. It is defined as:

$$\begin{pmatrix} [\tau] \\ [x](t) \\ [y](t) \end{pmatrix} \xrightarrow{\mathcal{C}_{\text{offset}}} \begin{pmatrix} [\tau] \cap \bigcap_{t_1=t_0}^{t_f} \left([y]^{-1}\left([x]\left(t_1\right)\right) - t_1 \right) \cap \bigcap_{t_1=t_0}^{t_f} \left(t_1 - [x]^{-1}\left([y]\left(t_1\right)\right) \right) \\ [x](t) \cap [y]\left(t + [\tau]\right) \\ [y](t) \cap [x]\left(t - [\tau]\right) \end{pmatrix}, \tag{6.8}$$

where $[t_0, t_f]$ is the time interval over which both $[x](\cdot)$ and $[y](\cdot)$ are defined.

Proof. We prove that $\mathcal{C}_{\text{offset}}$ is a contractor by showing that it satisfies both the contraction and the consistency property depicted in Section 3.7.2. This ensures that the contractor cannot enlarge the result and never loses a part of the solution.

Consistency property

We need to prove that no valid solution for neither τ, $x(\cdot)$ nor $y(\cdot)$ is lost. First, we prove that no valid solution for τ is lost. In order to do that, we prove the consistency of all three parts of the contraction function that are intersected, i.e. we prove consistency for

$$\begin{aligned} &(i) && [\tau] \\ &(ii) && \bigcap_{t_1=t_0}^{t_f} \left([y]^{-1} \left([x] \left(t_1 \right) \right) - t_1 \right) \\ &(iii) && \bigcap_{t_1=t_0}^{t_f} \left(t_1 - [x]^{-1} \left([y] \left(t_1 \right) \right) \right) \end{aligned} \tag{6.9}$$

The proof for (i) is trivial as $\tau \in [\tau]$.

Let $x(\cdot) \in [x](\cdot)$, $y(\cdot) \in [y](\cdot)$ and $\tau \in [\tau]$ such that $\forall t \in [t_0, t_f] : x(t) = y(t+\tau)$. We reformulate (ii) as

$$\begin{aligned} \bigcap_{t_1=t_0}^{t_f} \left([y]^{-1} \left([x] \left(t_1 \right) \right) - t_1 \right) &\supseteq \bigcap_{t_1=t_0}^{t_f} \left([y]^{-1} \left(x \left(t_1 \right) \right) - t_1 \right) \\ &= \bigcap_{t_1=t_0}^{t_f} \left([y]^{-1} \left(y \left(t_1 + \tau \right) \right) - t_1 \right) \\ &\supseteq \bigcap_{t_1=t_0}^{t_f} \left(y^{-1} \left(y \left(t_1 + \tau \right) \right) - t_1 \right) \\ &= \bigcap_{t_1=t_0}^{t_f} \left(t_1 + \tau - t_1 \right) \\ &= \bigcap_{t_1=t_0}^{t_f} \left(\tau \right) = \tau. \end{aligned} \tag{6.10}$$

Thus, we have proven that $\tau \in \left(\bigcap_{t_1=t_0}^{t_f} \left([y]^{-1} \left([x] \left(t_1 \right) \right) - t_1 \right) \right)$.

Next, the constraint $x(t) = y(t+\tau)$ can be reformulated as $x(t-\tau) = y(t)$. Using this, we can reformulate (iii) as

$$\begin{aligned} \bigcap_{t_1=t_0}^{t_f} \left(t_1 - [x]^{-1} \left([y] \left(t_1 \right) \right) \right) &\supseteq \bigcap_{t_1=t_0}^{t_f} \left(t_1 - [x]^{-1} \left(y \left(t_1 \right) \right) \right) \\ &= \bigcap_{t_1=t_0}^{t_f} \left(t_1 - [x]^{-1} \left(x \left(t_1 - \tau \right) \right) \right) \\ &\supseteq \bigcap_{t_1=t_0}^{t_f} \left(t_1 - x^{-1} \left(x \left(t_1 - \tau \right) \right) \right) \\ &= \bigcap_{t_1=t_0}^{t_f} \left(t_1 - \left(t_1 - \tau \right) \right) \\ &= \bigcap_{t_1=t_0}^{t_f} \left(\tau \right) = \tau. \end{aligned} \tag{6.11}$$

Thus, we have proven that $\tau \in \left(\bigcap_{t_1=t_0}^{t_f} \left(t_1 - [x]^{-1}\left([y]\left(t_1\right)\right)\right)\right)$, and therefore we have proven the consistency property for τ.

To prove the consistency property for $x(\cdot)$, according to the contractor definition in (6.8), we need to prove that $\forall t \in [t_0, t_f] : x(t) \in ([x](t) \cap [y]\left(t + [\tau]\right))$. Since $x(t) \in [x](t)$ is trivial, we only need to prove that $x(t) \in ([y]\left(t + [\tau]\right))$. The following reformulation proves this property:

$$[y]\left(t + [\tau]\right) \supseteq [y]\left(t + \tau\right) \supseteq y\left(t + \tau\right) = x\left(t\right) \tag{6.12}$$

Analogously, the consistency property for $y(\cdot)$ can be proven by using $x(t - \tau) = y(t)$. We omit the full expression at this point.

Contraction property

The proof is trivial since all three variables are intersected with themselves during the contraction, and thus their size cannot increase.

Finally, we have proven Proposition 1, and thus $\mathcal{C}_{\text{offset}}$ is indeed a contractor. ■

6.1.2.1 Algorithm

Since the contractor $\mathcal{C}_{\text{offset}}$ in (6.8) is defined over tubes that are continuous in time, it cannot directly work on tubes that are built from real data. As introduced in Section 3.10.1, such tubes are implemented using slices of identical width. Fig. 6.4 shows two tubes $[x](\cdot)$ and $[y](\cdot)$ that are represented by slices.

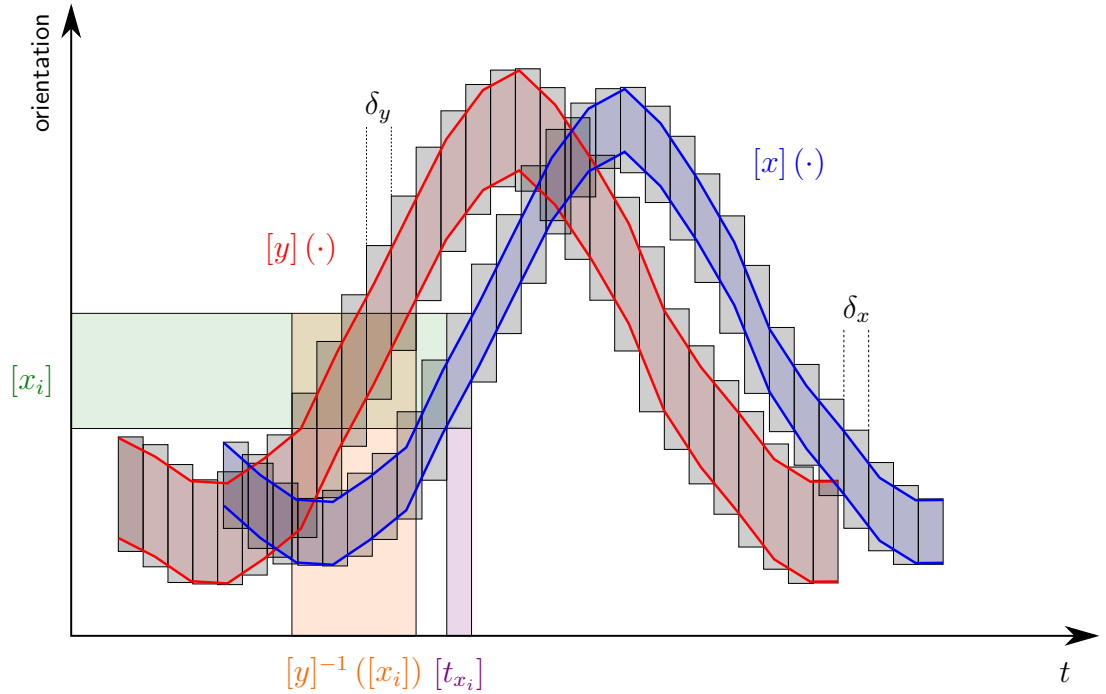


Figure 6.4: Exemplary tubes $[x](\cdot)$ and $[y](\cdot)$ which are shifted by a constant offset. This visualization serves to help understand the implementation of the time offset contractor $\mathcal{C}_{\text{offset}}$, which is detailed in Algorithm 4.

Consequently, Algorithm 4 depicts the algorithm that operates on these slices to contract the domain for the time offset $[\tau]$ as well as the envelope of both tubes $[x](\cdot)$ and $[y](\cdot)$. We explain it using the visualization in Fig. 6.4. At first, the algorithm iterates over the first tube $[x](\cdot)$ and extracts the time domain of each slice which we denote as $[t_{x_i}]$ (depicted in purple). Since the width of a slice is known to be δ_x this time domain can be computed straightforwardly. Subsequently, we are interested in finding the interval $[t_{y_i}]$ that constitutes the time domain over which the other tube $[y](\cdot)$ intersects $[x]([t_{y_i}])$. However, the tube $[y](\cdot)$ may not be defined over this time domain. Thus, we have to check first if $[t_{\tau_i}]$ is a true subset of $[t_y]$, where $[t_y]$ is the time domain over which $[y](\cdot)$ is defined. Afterwards, we compute the image $[x_i]$ (depicted in green) of the tube $[x](\cdot)$ during the time domain $[t_{x_i}]$. Next, the tube $[y](\cdot)$ is inverted to determine the desired time domain $[t_{y_i}]$ (depicted in orange). Finally, the definition of the contractor given in (6.8) is applied in lines 7 and 8. As the algorithm iterates over $[x](\cdot)$, only the corresponding slices of this tube are contracted here. Subsequently, the algorithm iterates over the second tube $[y](\cdot)$ and repeats the same procedure.

Algorithm 4: Time offset contractor $\mathcal{C}_{\text{offset}}$

input : $[\tau]$, $[x](\cdot)$, $[y](\cdot)$
output : $[\tau]$, $[x](\cdot)$, $[y](\cdot)$

1 **for** $i = 0$ **to** n_x **do** // iterate over the first tube $[x](\cdot)$
2 $[t_{x_i}] \coloneqq [i \cdot \delta_x, (i+1) \cdot \delta_x]$; // δ_x is the width of a slice of the tube $[x](\cdot)$
3 $[t_{\tau_i}] \coloneqq [t_{x_i}] + [\tau]$;
4 **if** $[t_{\tau_i}] \subset [t_y]$ **then** // $[t_y]$ is the time interval over which $[y](\cdot)$ is defined
5 $[x_i] \coloneqq [x]([t_{x_i}])$; // interval evaluation of a tube
6 $[t_{y_i}] \coloneqq [y]^{-1}([x_i], [t_{\tau_i}])$; // tube inversion
7 $[\tau] \coloneqq [\tau] \cap ([t_{y_i}] - [t_{x_i}])$; // contraction of $[\tau]$
8 $[x]([t_{x_i}]) \coloneqq [x]([t_{x_i}]) \cap ([y]([t_{x_i}] + [\tau]))$; // contraction of a slice of $[x](\cdot)$
9 **end**
10 **end**
11 **for** $i = 0$ **to** n_y **do** // iterate over the second tube $[y](\cdot)$
12 $[t_{y_i}] \coloneqq [i \cdot \delta_y, (i+1) \cdot \delta_y]$; // δ_y is the width of a slice of the tube $[y](\cdot)$
13 $[t_{\tau_i}] \coloneqq [t_{y_i}] - [\tau]$;
14 **if** $[t_{\tau_i}] \subset [t_x]$ **then** // $[t_x]$ is the time interval over which $[x](\cdot)$ is defined
15 $[y_i] \coloneqq [y]([t_{y_i}])$; // interval evaluation of a tube
16 $[t_{x_i}] \coloneqq [x]^{-1}([y_i], [t_{\tau_i}])$; // tube inversion
17 $[\tau] \coloneqq [\tau] \cap ([t_{y_i}] - [t_{x_i}])$; // contraction of $[\tau]$
18 $[y]([t_{y_i}]) \coloneqq [y]([t_{y_i}]) \cap ([x]([t_{y_i}] - [\tau]))$; // contraction of a slice of $[y](\cdot)$
19 **end**
20 **end**

Remark

When iterating over the tubes in Algorithm 4, each slice corresponds to a single contractor for the time offset $[\tau]$. However, not all such contractors are equally useful to contract the time offset and some result in no contraction due to weak constraints (e.g. if the neighboring slices enclose a similar domain, i.e. if the derivative is close to zero). In order to design a more efficient algorithm, not all contractors have to be taken into account, but a small subset that ideally leads to the same contraction. However, this is not the focus of this thesis and remains a question for future work.

6.1.3 Spatiotemporal calibration

After introducing the time offset contractor in the previous section, we couple it with SIVIA to compute the extrinsic transformation $\mathbf{R}_C^I$ and the offset τ simultaneously in this section. This approach was previously published in [27].

First, we need to reduce the number of variables of $\mathbf{R}_C^I$ by expressing it using a different rotation representation (cf. Section 2.3). Inspired by [38], we use the MRP since they exhibit several advantages. First, to express an inverse rotation it suffices to negate the three parameters. This allows us to reformulate (6.6) as

$$\begin{aligned} \forall t: \ \boldsymbol{\rho}_C^{C_0}(t) &= -\boldsymbol{\rho}_C^I \bullet \boldsymbol{\rho}_I^{I_0}(t+\tau) \bullet \boldsymbol{\rho}_C^I, \\ \boldsymbol{\rho}_C^{C_0}(\cdot) \in [\boldsymbol{\rho}_C^{C_0}](\cdot), \boldsymbol{\rho}_I^{I_0}(\cdot) &\in [\boldsymbol{\rho}_I^{I_0}](\cdot), \boldsymbol{\rho}_C^I \in [\boldsymbol{\rho}_C^I], \tau \in [\tau], \end{aligned} \tag{6.13}$$

where $\boldsymbol{\rho}_B^A$ is the three-dimensional MRP vector for the rotation from coordinate system A to B, i.e. the equivalent to the rotation matrix $\mathbf{R}_B^A$. Furthermore, $(\bullet)$ is the operator defining a sequential rotation (cf. Section 2.3.2).

Being able to compose two orientations straightforwardly using the $(\bullet)$ operator constitutes the second advantage of the MRP. Combining these two advantages allows us to drastically reduce the number of occurrences of the elements of the MRP vector $\mathbf{b}$ in the expression

$$-\mathbf{a} \bullet \mathbf{b} \bullet \mathbf{a}, \tag{6.14}$$

where $\mathbf{a}$ is another MRP vector. In total, each component of $\mathbf{b}$ appears only once in each row of (6.14). This allows us to find a tighter enclosure for the right side of (6.13). Due to its size, we omit the full expression.

The third advantage of the MRP is that the singular points are as far away from the origin as possible, which allows us to perform drastic rotations of our sensor setup during the experiment before encountering singularities.

Finally, we depict the whole process to compute domains enclosing the spatiotemporal calibration parameters $\boldsymbol{\rho}_C^I$ and τ such that they satisfy (6.13). At first, we position the rigidly mounted sensor setup in front of the calibration target to achieve a common starting point t_0 for both sensors. Subsequently, the sensor setup is rotated either manually or by using a motor such that the calibration target remains in view of the camera over the whole experiment.

Now, we compute the three-dimensional tube enclosing the rotation of the IMU from the common starting point t_0: $[\boldsymbol{\rho}_I^{I_0}](\cdot)$. In order to do that, we apply our bounded-error model to the angular velocities measured by the IMU and integrate them under interval uncertainty (cf. Section 4.2.3). Likewise, we determine the rotation of the camera: $[\boldsymbol{\rho}_C^{C_0}](\cdot)$. For this, we model the uncertainties of the detected checkerboard corners and their respective 3D coordinates in the checkerboard coordinate system using intervals (cf. Section 4.2.2). Subsequently, we solve the PnP problem under interval uncertainty (cf. Chapter 5) which results in the orientation of the camera in the world coordinate system. Thus, we apply (6.5) to determine the rotation from the common starting point.

From here, Algorithm 5 depicts the further procedure. It follows the standard SIVIA algorithm, except that only the extrinsic rotation parameters are bisected in line 11. Furthermore, the

Algorithm 5: IMU-Camera calibration

input : $[\tau]$, $[\boldsymbol{\rho}_C^I]$, $[\boldsymbol{\rho}_I^{I_0}](\cdot)$, $[\boldsymbol{\rho}_C^{C_0}](\cdot)$, ϵ
output : $\overline{\mathbb{X}}$, which is a subpaving for the offset τ and the 3 DOF rotation $\boldsymbol{\rho}_C^I$

1 $\mathcal{S} \coloneqq \left\{\left\{[\tau], [\boldsymbol{\rho}_C^I]\right\}\right\}$; // $\mathcal{S}$ is a stack
2 **while** $\mathcal{S} \neq \varnothing$ **do**
3 $\quad \left\{[\tau], [\boldsymbol{\rho}_C^I]\right\} \coloneqq \mathsf{top}(\mathcal{S})$;
$\quad$ // The IMU tube is rotated into the camera coordinate system
4 $\quad [\boldsymbol{\rho'}_I^{I_0}](\cdot) \coloneqq -[\boldsymbol{\rho}_C^I] \bullet [\boldsymbol{\rho}_I^{I_0}](\cdot) \bullet [\boldsymbol{\rho}_C^I]$;
$\quad$ // $\mathcal{C}_{\mathtt{offset}}$ is called separately for all three dimensions of the tubes
5 $\quad \mathcal{C}_{\mathsf{offset}}\left([\tau], [\boldsymbol{\rho}_C^{C_0}](\cdot), [\boldsymbol{\rho'}_I^{I_0}](\cdot)\right)$;
6 $\quad$ **if** *any of* $\left\{[\tau], [\boldsymbol{\rho}_C^{C_0}](\cdot), [\boldsymbol{\rho'}_I^{I_0}](\cdot)\right\} = \varnothing$ **then**
7 $\quad\quad$ continue;
8 $\quad$ **else if** $w\left([\boldsymbol{\rho}_C^I]\right) < \epsilon$ **then**
9 $\quad\quad \overline{\mathbb{X}} \coloneqq \overline{\mathbb{X}} \cup \left\{[\tau], [\boldsymbol{\rho}_C^I]\right\}$;
10 $\quad$ **else**
$\quad\quad$ // only the extrinsic rotation parameters are bisected
11 $\quad\quad \left([\boldsymbol{\rho}_{C,1}^I], [\boldsymbol{\rho}_{C,2}^I]\right) \coloneqq \mathsf{bisect}\left([\boldsymbol{\rho}_C^I]\right)$;
12 $\quad\quad \mathcal{S} \coloneqq \mathcal{S} \cup \left\{\left\{[\tau], [\boldsymbol{\rho}_{C,1}^I]\right\}\right\} \cup \left\{\left\{[\tau], [\boldsymbol{\rho}_{C,2}^I]\right\}\right\}$;
13 $\quad$ **end**
14 **end**

rotation parameters of the current iteration are applied to the IMU tube in line 4 to transform it into the camera coordinate system according to (6.13). Subsequently, in line 5 the previously introduced time offset contractor $\mathcal{C}_{\mathsf{offset}}$ is called for the properly rotated tubes to contract $[\tau]$. Since the tubes are three-dimensional, the contractor must be called separately for each row. Finally, the algorithm outputs a subpaving for both the extrinsic rotation parameters $\boldsymbol{\rho}_C^I$ and the time offset τ.

6.2 Extrinsic calibration between camera and LiDAR

The following approach was previously published in [113].

To fuse data from camera and laser scanner, we need to be able to transform laser scan points into the camera coordinate system, or vice versa. In order to do so, it is inevitable to know the extrinsic transformation parameters between camera and laser scanner. These transformation parameters consist of the rotation matrix $\mathbf{R}_L^C$ and the translation vector $\mathbf{T}_L^C$ and allow us to establish a link the sensor coordinate systems as follows:

$$\mathbf{X}^C = \mathbf{R}_L^C \cdot \mathbf{X}^L + \mathbf{T}_L^C, \tag{6.15}$$

where $\mathbf{X}^C$ and $\mathbf{X}^L$ are 3D points in the camera and laser scanner coordinate system, respectively. The process of computing $\mathbf{R}_L^C$ and $\mathbf{T}_L^C$ is denoted as *extrinsic calibration*.

Building on the sensor error models introduced in Chapter 4, we present an approach to compute $\mathbf{R}_L^C$ and $\mathbf{T}_L^C$ in a bounded-error context. Instead of finding only point-valued results for the transformation parameters, we compute intervals which are guaranteed to enclose the true parameters - naturally only if the initial assumption about bounded sensor errors is valid. Thus, our extrinsic calibration approach can be described as *guaranteed*, *consistent* and *reliable*. Furthermore, interval analysis allows us to not only enclose the transformation parameters, but also to assess their uncertainty by evaluating the interval widths. This poses an advantage over conventional methods which can often only specify the accuracy of the transformation parameters by comparing them to ground truth information (cf. Section 2.5.2).

In addition to the consistency and the possibility to assess the uncertainty, the proposed approach does not need good initial values to find the correct solution. Unlike optimization methods that can converge to a local minimum, interval analysis allows us to compute intervals guaranteed to contain the global minimum, starting from an arbitrarily large domain.

The general idea to perform the extrinsic calibration is to find features in both camera and laser scan data which can be associated automatically. Inspired by the method introduced by Zhou et al. [49], we find plane, line and point correspondences from a checkerboard which constrain the rigid body transformation between the sensor coordinate systems. Fig. 6.5 visualizes the idea. The feature extraction process is detailed for the camera in Section 6.2.1 and for the laser scanner in Section 6.2.2. Subsequently, in Section 6.2.3 the plane, line and point correspondences are used to formulate the CSP which is then solved using forward-backward contractors and SIVIA.

Although it is possible to perform the extrinsic calibration using one single camera image and laser scan, we generally employ multiple checkerboard poses. This means that we position the checkerboard differently in front of the sensors for each recording. The reason for this is that a single checkerboard might not constrain all transformation parameters adequately. For example, one checkerboard pose may impose strong constraints on the rotation around the z-axis, but is not suitable to contract the translation along the x-axis. Thus, it is necessary to combine constraints enforced by several checkerboard poses. However, the following explanation of our method considers one checkerboard pose only. Nevertheless, multiple poses can be combined by merging the resulting contractors in the final SIVIA procedure.

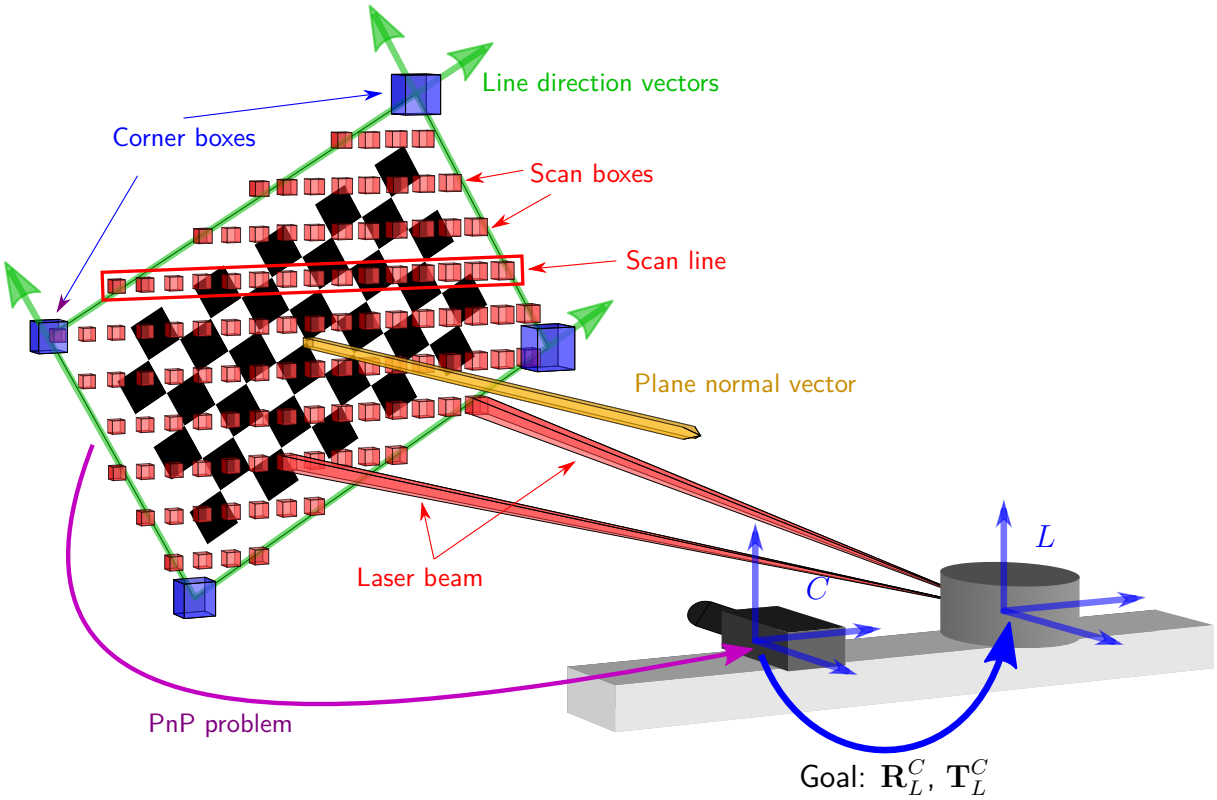


Figure 6.5: To find the extrinsic transformation, i.e. the rotation matrix $\mathbf{R}_L^C$ and the translation vector $\mathbf{T}_L^C$, between camera and laser scanner in a guaranteed way we find plane, line and point features in both camera and laser scan data under interval uncertainty.

6.2.1 Camera feature extraction

We aim to find three-dimensional plane, line and point features on the checkerboard in a single camera image. Fig. 6.5 visualizes the features we are interested in. Since the rigid body transformation between checkerboard and camera coordinate system is unknown, we have to first solve the PnP problem as detailed in Chapter 5. This results in interval domains for the translation vector $\mathbf{T}_W^C \in [\mathbf{T}_W^C]$ and the rotation matrix $\mathbf{R}_W^C \in [\mathbf{R}_W^C]$, which define the transformation between the checkerboard (or world) coordinate system W and the camera coordinate system C.

Plane feature extraction

First, we are interested in finding the checkerboard plane equation as observed from the camera coordinate system. The general plane equation is

$$ax + by + cz + d = \mathbf{n} \cdot \mathbf{X} + d = 0, \tag{6.16}$$

where $\mathbf{n} = (a\ b\ c)^\intercal$ is the plane normal vector, $\mathbf{X} = (x\ y\ z)^\intercal$ is any 3D point on the plane and d is a constant. Our checkerboard coordinate system W is defined such that the plane intersects the origin and $\mathbf{n}^W = (0\ 0\ 1)^\intercal$ is the normal vector of the plane. Thus, to determine the plane normal vector in the camera coordinate system, we have to transform $\mathbf{n}^W$ using the previously determined rigid body transformation. Since $\mathbf{n}^W$ is a vector, and thus the

transformation is independent of the translation, we can compute the plane normal using solely the rotation matrix as

$$\begin{aligned} \mathbf{n}^C &= \mathbf{R}_W^C \cdot \mathbf{n}^W = \mathbf{r}_3, \\ \mathbf{n}^C \in [\mathbf{n}^C], \mathbf{R}_W^C &\in [\mathbf{R}_W^C], \mathbf{r}_3 \in [\mathbf{r}_3], \end{aligned} \tag{6.17}$$

where $\mathbf{r}_3$ is the third column of the rotation matrix $\mathbf{R}_W^C$. Subsequently, any point $\mathbf{X}_0^W$ belonging to the checkerboard plane that is transformed into the camera coordinate system can be used to compute d^C. We choose $\mathbf{X}_0^W = (\,0 \;\; 0 \;\; 0\,)^\intercal$, and thus

$$\begin{aligned} d^C &= -(\mathbf{n}^C \cdot \mathbf{X}_0^C) \\ &= -(\mathbf{n}^C \cdot (\mathbf{R}_W^C \cdot \mathbf{X}_0^W + \mathbf{T}_W^C)) \\ &= -(\mathbf{n}^C \cdot \mathbf{T}_W^C), \\ d^C \in [d^C], \mathbf{n}^C &\in [\mathbf{n}^C], \mathbf{T}_W^C \in [\mathbf{T}_W^C]. \end{aligned} \tag{6.18}$$

Point feature extraction

Second, taking advantage of the known checkerboard dimensions, we are able to immediately determine the four corner points $\mathbf{C}_m^W$, $m \in \{1, \ldots, 4\}$, in the checkerboard coordinate system. Subsequently, they are transformed into the camera coordinate system using the previously determined rigid body transformation:

$$\begin{aligned} \mathbf{C}_m^C &= \mathbf{R}_W^C \cdot \mathbf{C}_m^W + \mathbf{T}_W^C, \\ \mathbf{C}_m^C \in [\mathbf{C}_m^C], \mathbf{R}_W^C &\in [\mathbf{R}_W^C], \mathbf{C}_m^W \in [\mathbf{C}_m^W], \mathbf{T}_W^C \in [\mathbf{T}_W^C]. \end{aligned} \tag{6.19}$$

with $m \in \{1, \ldots, 4\}$ and

$$[\mathbf{C}_m^W] = \mathbf{C}_m^W + [\mathbf{\Delta}_W], \tag{6.20}$$

where $[\mathbf{\Delta}_W]$ is the accuracy of the world points of the checkerboard that is given by the manufacturing accuracy of the checkerboard.

Line feature extraction

Third, we aim to find the line equations of the four border lines surrounding the checkerboard. The general line equation we use is

$$\mathbf{X} = \mathbf{X}_0 + r\,\mathbf{d}, \tag{6.21}$$

where $\mathbf{X}_0$ is the starting point of the line, r is a constant, $\mathbf{d}$ is the (unit) direction vector of the line and $\mathbf{X}$ is any point on the line. Since we already know two points belonging to each line, namely the corner points previously computed, only the direction vector is left to be determined. Reformulating (6.21) yields that an arbitrarily scaled direction vector can be computed by subtracting two points from one another. Let $\mathbf{C}_j^W = (\,x_j^W \;\; y_j^W \;\; 0\,)^\intercal$ and $\mathbf{C}_k^W = (\,x_k^W \;\; y_k^W \;\; 0\,)^\intercal$ be two adjacent corner points given in the checkerboard coordinate system as depicted in Fig. 6.5. Computing $\mathbf{d}_i^C$ means transforming the two corner points into

the camera coordinate system and then subtracting them while exploiting the z-coordinate to be 0:

$$\begin{aligned} r_i\,\mathbf{d}_i^C &= (\mathbf{R}_W^C \cdot \mathbf{C}_j^W + \mathbf{T}_W^C) - (\mathbf{R}_W^C \cdot \mathbf{C}_k^W + \mathbf{T}_W^C) \\ &= \mathbf{R}_W^C \cdot (\mathbf{C}_j^W - \mathbf{C}_k^W) \\ &= \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{pmatrix} \cdot \hat{\mathbf{d}}_i^W, \end{aligned} \tag{6.22}$$

where $\mathbf{r}_l$, $l = \{1, 2\}$, is the l-th column of the rotation matrix $\mathbf{R}_W^C$ and

$$\begin{aligned} \hat{\mathbf{d}}_i^W &\coloneqq \begin{pmatrix} x_j^W - x_k^W \\ y_j^W - y_k^W \end{pmatrix}, \\ \hat{\mathbf{d}}_i^W \in [\hat{\mathbf{d}}_i^W], \begin{pmatrix} x_{\{j,k\}}^W \\ y_{\{j,k\}}^W \end{pmatrix} &\in \begin{pmatrix} [x_{\{j,k\}}^W] \\ [y_{\{j,k\}}^W] \end{pmatrix}, \end{aligned} \tag{6.23}$$

with $i, j, k \in \{1, \ldots, 4\}$ and

$$\begin{pmatrix} [x_{\{j,k\}}^W] \\ [y_{\{j,k\}}^W] \end{pmatrix} = \begin{pmatrix} x_{\{j,k\}}^W + [\Delta_W] \\ y_{\{j,k\}}^W + [\Delta_W] \end{pmatrix}, \tag{6.24}$$

where $[\Delta_W]$ is the accuracy of the world points of the checkerboard.

To compute the unit vector that is not scaled by r_i, it is convenient to normalize the direction vector first since rotating it afterwards preserves its length:

$$\begin{aligned} \mathbf{d}_i^C &= \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{pmatrix} \cdot \frac{\hat{\mathbf{d}}_i^W}{\|\hat{\mathbf{d}}_i^W\|}, \\ \mathbf{d}_i^C \in [\mathbf{d}_i^C], \mathbf{r}_1 \in [\mathbf{r}_1], \mathbf{r}_2 &\in [\mathbf{r}_2], \hat{\mathbf{d}}_i^W \in [\hat{\mathbf{d}}_i^W] \end{aligned} \tag{6.25}$$

6.2.2 Laser scanner feature extraction

As for the camera data, we extract the same plane, line and point features from laser scan data. However, identifying these features in point clouds is more difficult since the transformation between checkerboard and sensor coordinate system cannot be calculated as conveniently. In addition, using intensity data to detect the the checkerboard features as fixed points for the checkerboard coordinate system is impractical, if intensity measurements are not available.

Before we can compute the plane parameters from the laser scan points, we have to identify all points belonging to the plane of the board. In order to do that, we make use of the dedicated environment in which the extrinsic calibration is performed - i.e. a laboratory which is set up to not contain any objects around the checkerboard. Thus, any scan point residing in a pre-defined area is said to belong to the checkerboard. As will be explained later, outliers (e.g. on the checkerboard boundaries) can be taken into account by employing a q-relaxed intersection (cf. Section 3.9). Let N_p be the number of interval points residing on the plane and $[\mathbf{P}_l^L]$, $l \in \{1, \ldots, N_p\}$, an interval point according to the bounded error model introduced in Chapter 4. Fig. 6.5 shows exemplary interval points on the checkerboard plane and the corresponding plane, line and point features we are striving for.

Plane feature extraction

We employ the general plane equation (6.16) to formulate a CSP for the plane normal vector. Additionally, we introduce a second constraint to restrict the normal vector to a unit vector. The CSP $\mathcal{P}$ can be formulated as:

$$\mathcal{P}: \begin{cases} \textbf{Variables: } \mathbf{n}^L,\, d^L,\, \mathbf{P}_l^L \\ \textbf{Constraints:} \\ \quad 1.\ \ \mathbf{n}^L \cdot \mathbf{P}_l^L + d^L = 0 \qquad \text{for } l \in \{1, \ldots, N_p\} \\ \quad 2.\ \ \|\mathbf{n}^L\| = 1 \\ \textbf{Domains: } [\mathbf{n}^L], [d^L], [\mathbf{P}_l^L] \end{cases} \tag{6.26}$$

The domains are initialized with $[\mathbf{n}^L] = ([-1, 1]_{\times 3})^\intercal$ and $[d^L] = [0, \infty]$. In words, we consider the normal vector to be unknown, but restrict $d^L \geq 0$ to remove the ambiguity of positive/negative unit vectors. To solve the CSP, we employ a forward-backward contractor (cf. Section 3.7.2) in conjunction with the SIVIA algorithm (cf. Section 3.8). SIVIA is necessary since there are $N_p + 1$ constraints in total (one for each point on the plane and the unit vector constraint) each involving the normal vector $\mathbf{n}^L$. To reduce complexity, we refrain from bisecting d^L and only bisect $\mathbf{n}^L$.

Besides the forward-backward contractor, it is possible to run a preconditioned Gauss-Seidel contractor [9]. However, this contractor works only for linear square systems. Since our system is not linear due to the second constraint and generally not square due to $N_p > 4$ (i.e. there are more scan points on the plane than unknown variables), it is not possible to use solely this contractor to achieve a satisfying contraction. Nevertheless, it can be used in conjunction with the forward-backward contractor by randomly selection four points to formulate a square linear system based on the first constraint.

While contracting the domains $[\mathbf{n}^L]$ and $[d^L]$ in every step of the SIVIA algorithm, we simultaneously contract the domains for each point $[\mathbf{P}_l^L]$. This allows us to reduce the uncertainty of our interval points which will be further needed to determine the line and point features.

In the presence of outliers, it is possible to employ a q-relaxed intersection for the first constraint of the CSP $\mathcal{P}$. Instead of requiring that all points have to fulfill this constraint, we allow a pre-determined number of points, which depends on the sensor and the environment, to be potential outliers. Afterwards, we detect guaranteed outliers by checking for every point whether the constraints of the CSP $\mathcal{P}$ uphold with the contracted domains for $[\mathbf{n}^L]$ and $[d^L]$. If this is not the case, the relevant point is removed from the further feature extraction process.

Line feature extraction

The line feature extraction consists of two steps since the representation of a line $i \in \{1, \ldots, 4\}$ requires the line direction vector $\mathbf{d}_i^L$ and at least one point $\mathbf{Q}_{ij}^L$, with $j \in \{1, \ldots, N_i\}$, on the line. N_i is the total number of points we compute for the line i. At first, we determine so-called *hypothetical laser rays* originating from the laser scanner and passing through the border of the checkerboard. Subsequently, these hypothetical laser rays are intersected with

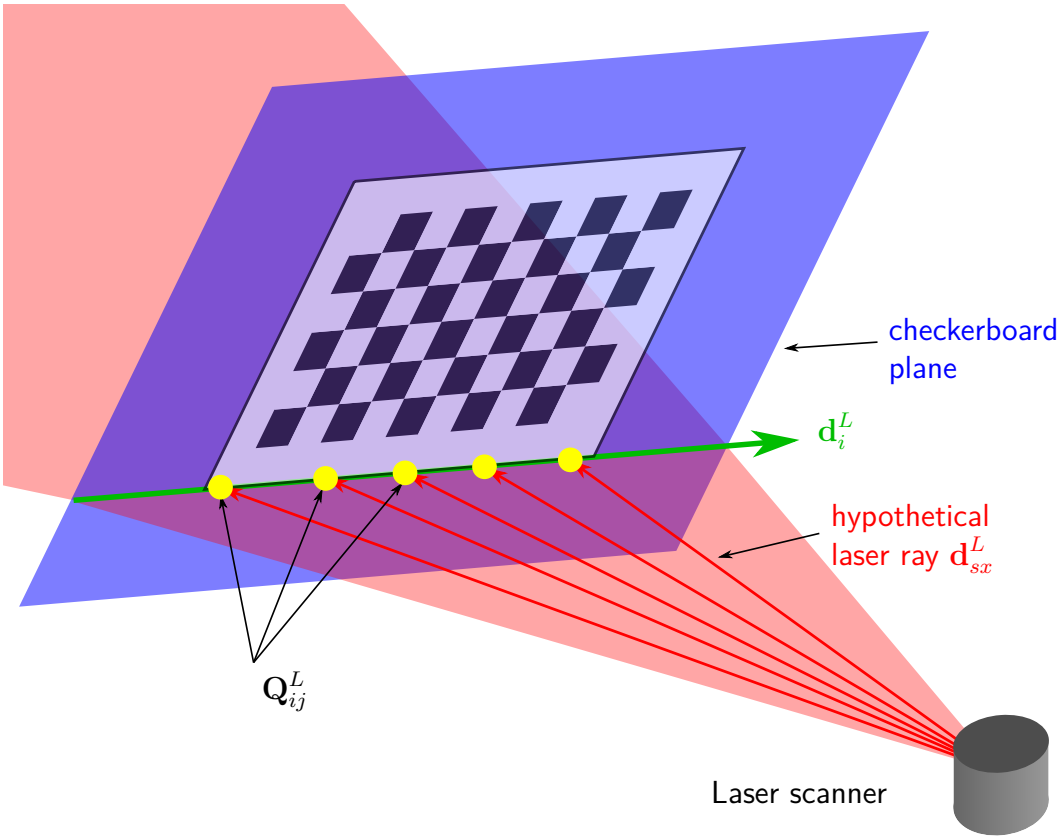


Figure 6.6: Visualization of the line feature extraction. Hypothetical laser rays (red) originating from the laser scanner and passing through the border of the checkerboard are intersected with the checkerboard plane (blue) to find the desired border points (yellow). Furthermore, the hypothetical laser rays span a plane (red), which is intersected with the checkerboard plane to find the direction vector of the border line.

the checkerboard plane to find the line points $\mathbf{Q}_{ij}^L$. Moreover, these hypothetical rays span another plane, which is intersected with the checkerboard plane to find the direction vector of the border line $\mathbf{d}_i^L$. Fig. 6.6 visualizes the line feature extraction.

We explain the process of determining the hypothetical laser rays by looking at a single scan line $s \in \{1, \ldots, N_s\}$, where N_s is the total number of scan lines (cf. Fig. 6.5). Fig. 6.7 shows such a scan line in top view. Let $\mathbf{P}_{sm}^L$ be a point belonging to this scan line with $m = \{1, \ldots, n_s\}$, where n_s is the total number of points on the scan line s. We know that all points of this scan line s have an overlapping interval for the vertical opening angle

$$[\theta_s] = \bigcap_{m=\{1,\ldots,n_s\}} [\theta_{sm}], \tag{6.27}$$

where $\theta_{sm} \in [\theta_{sm}]$ is the vertical opening angle of $\mathbf{P}_{sm}^L$ according to the bounded error model introduced in Chapter 4. However, the horizontal opening angles $\varphi_{sm} \in [\varphi_{sm}]$ are different.

Now, the idea is to find the interval for the horizontal opening angle $[\varphi_{sx_c}]$ of the right/left-most point on the checkerboard plane and the interval for the horizontal opening angle $[\varphi_{sx_b}]$ of the adjacent scan point that did not hit the plane but the background. $x \in \{r, l\}$ indicates

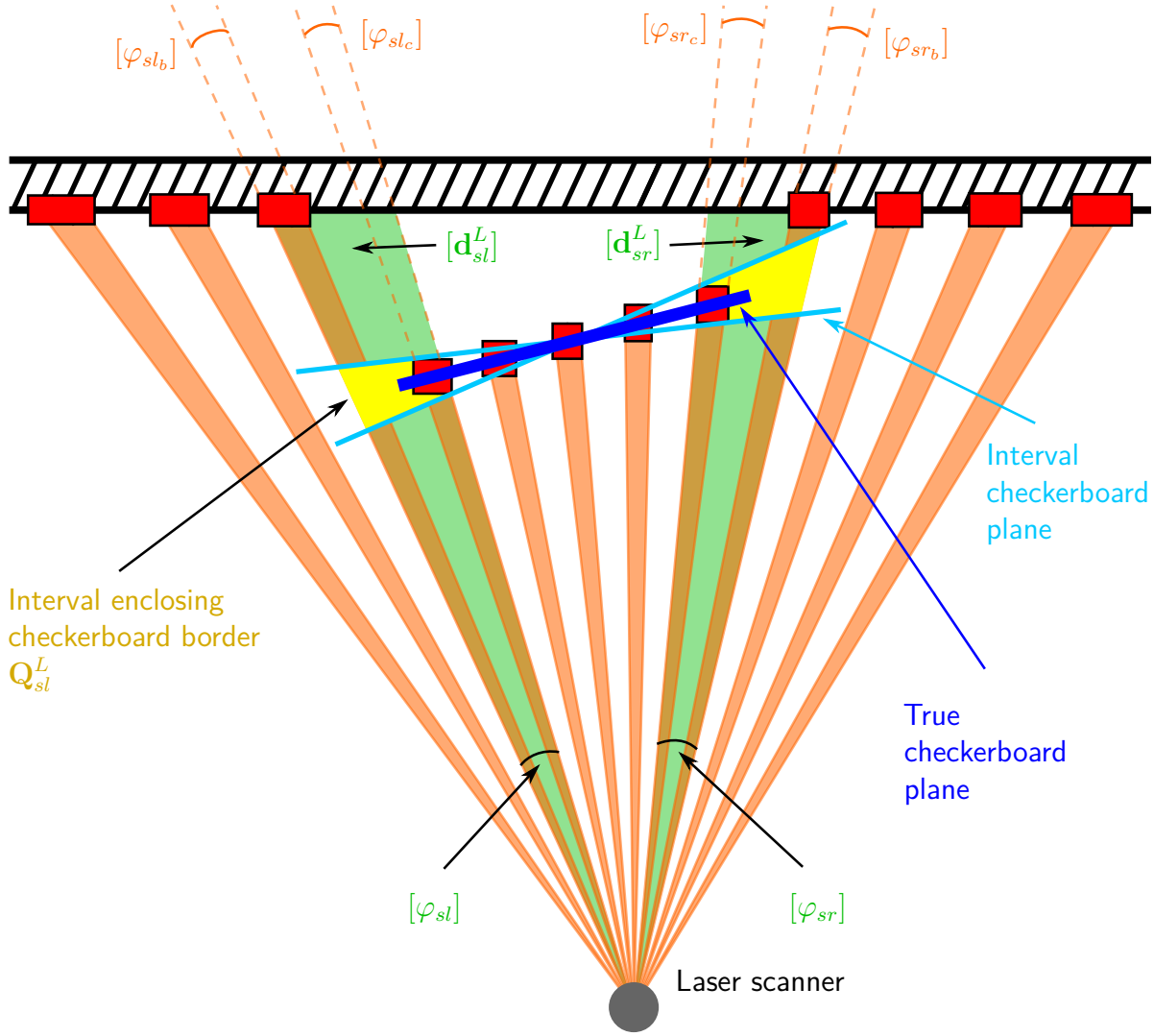


Figure 6.7: Top view of a single scan line s. The laser scanner measures several points (red) in the background and on the checkerboard. According to our bounded error model we assign an interval error to the horizontal opening angle of each measurement (orange rays). Consequently, we are able to find the horizontal opening angles $[\varphi_{sr}]$ and $[\varphi_{sl}]$ of the hypothetical laser rays passing through the borders of the checkerboard. We do so by computing the union over the last scan point residing on the checkerboard and the adjacent scan point hitting the background. Using these horizontal opening angles and the vertical opening angle, which is common among all scan points of the same scan line, we are able to determine the hypothetical laser rays $[\mathbf{d}^L_{sl}]$ and $[\mathbf{d}^L_{sr}]$. Finally, we compute a box enclosing the checkerboard border points by intersecting those laser rays with the previously determined interval checkerboard plane. The light blue lines depict both extremes of a plane containing all scan points.

whether we consider the right or left border of the checkerboard. This allows us to find a guaranteed interval for the horizontal opening angle of the desired hypothetical laser ray as

$$[\varphi_{sx}] = [\varphi_{sx_c}] \sqcup [\varphi_{sx_b}]. \tag{6.28}$$

Often, laser rays hitting both the checkerboard and the background are problematic since they result in two different reflections that correspond to two different distances. Generally, these distances are averaged by the laser scanner, resulting in a non-existent point between checkerboard and background. However, our approach can cope with such laser rays since we only require the horizontal opening angle, but not the distance.

Subsequently, we can state the direction vector of the hypothetical laser ray originating from the laser scanner and passing through the border of the checkerboard as

$$\begin{gathered} \mathbf{d}_{sx}^L = \begin{pmatrix} \sin\theta_s \cos\varphi_{sx} \\ \sin\theta_s \sin\varphi_{sx} \\ \cos\theta_s \end{pmatrix}, \\ \mathbf{d}_{sx}^L \in [\mathbf{d}_{sx}^L], \theta_s \in [\theta_s], \varphi_{sx} \in [\varphi_{sx}]. \end{gathered} \tag{6.29}$$

To find points on the border line of the checkerboard, we intersect the hypothetical laser rays with the checkerboard plane. Let $\mathbf{Q}_{sx}^L$ be the right/left border point we are looking for. To be on the hypothetical laser ray $\mathbf{d}_{sx}^L$ it must fulfill the equation

$$\mathbf{Q}_{sx}^L = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + r_{sx}\mathbf{d}_{sx}^L, \tag{6.30}$$

where r_{sx} is an arbitrary factor.
Furthermore, according to the previous paragraph, to lie on the checkerboard plane it must fulfill the equation

$$\mathbf{n}^L \cdot \mathbf{Q}_{sx}^L + d^L = 0. \tag{6.31}$$

Substituting (6.30) into (6.31) and solving for r_{sx} yields

$$r_{sx} = -\frac{d^L}{\mathbf{n}^L \cdot \mathbf{d}_{sx}^L}. \tag{6.32}$$

Thus, we can finally compute the box $[\mathbf{Q}_{sx}^L]$ enclosing the border point

$$\begin{gathered} \mathbf{Q}_{sx}^L = -\frac{d^L \mathbf{d}_{sx}^L}{\mathbf{n}^L \cdot \mathbf{d}_{sx}^L}, \\ \mathbf{Q}_{sx}^L \in [\mathbf{Q}_{sx}^L], d^L \in [d^L], \mathbf{d}_{sx}^L \in [\mathbf{d}_{sx}^L], \mathbf{n}^L \in [\mathbf{n}^L]. \end{gathered} \tag{6.33}$$

After computing points on the boundary lines of the checkerboard, it remains to determine the direction vectors of the lines. As previously explained and depicted in Fig. 6.6, we find the plane spanned by the hypothetical laser rays and intersect it with the checkerboard plane. Starting from $s = 1$, we iteratively add the hypothetical laser rays $[\mathbf{d}_{sx}^L]$ to the set $\mathbb{D}_{x_1}$ in

ascending order. Simultaneously, starting from $s = n_s$, we iteratively add the hypothetical laser rays $[\mathbf{d}_{sx}^L]$ to the set $\mathbb{D}_{x_2}$ in descending order. At every step we compute the plane normal of the plane spanned by all vectors in the set as

$$[\mathbf{n}_{x_m}^L] \coloneqq \bigcap_{\mathbf{d}_{jx}^L \in \mathbb{D}_{x_m}} \bigcap_{\substack{\mathbf{d}_{kx}^L \in \mathbb{D}_{x_m} \\ j \neq k}} \frac{[\mathbf{d}_{jx}^L] \times [\mathbf{d}_{kx}^L]}{\|[\mathbf{d}_{jx}^L] \times [\mathbf{d}_{kx}^L]\|}. \tag{6.34}$$

If $[\mathbf{n}_{x_m}^L]$ becomes empty, we know that the last inserted hypothetical laser ray does not lie in the common plane. Thus, it is removed again and the iteration stops. Afterwards, if a hypothetical laser ray belongs to both sets, it is omitted.

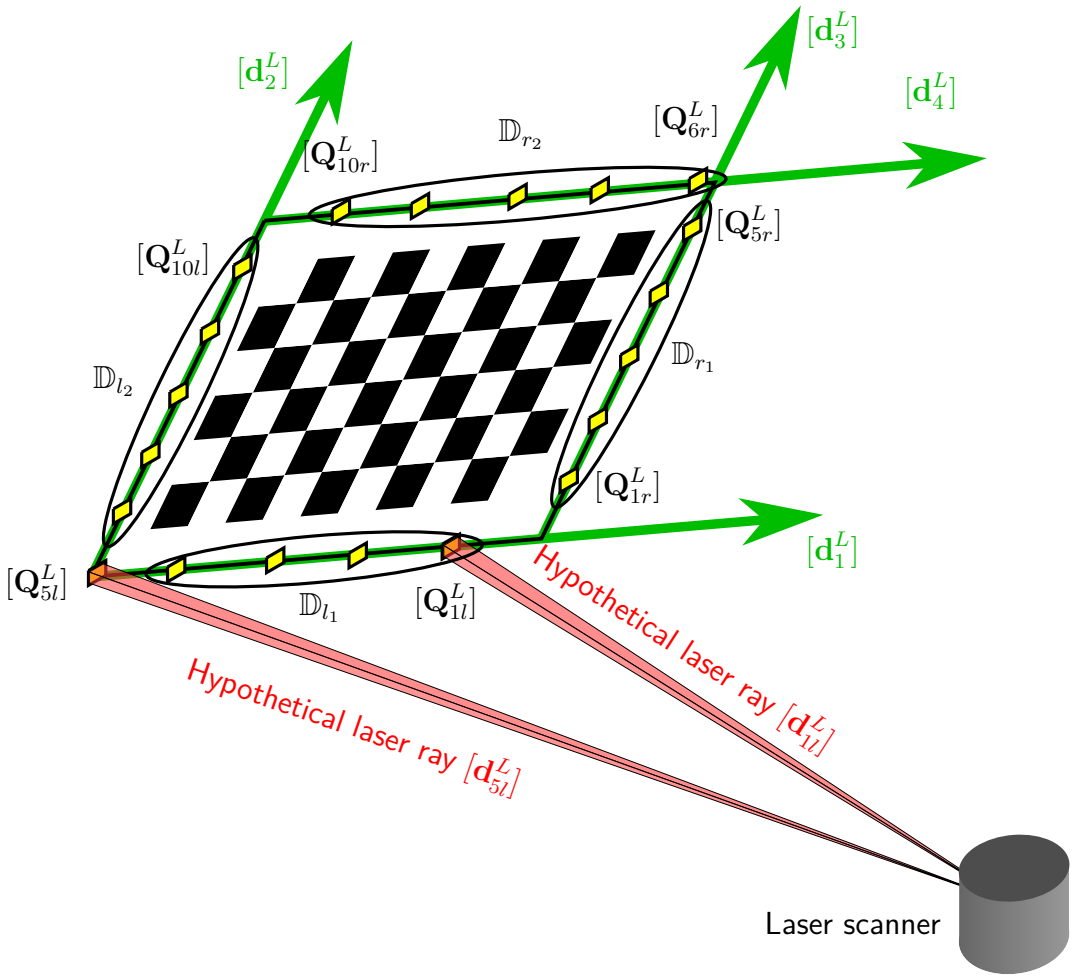


Figure 6.8: Visualization of the border points and corresponding boundary lines. As can be seen, border points $\mathbf{Q}_{sx}^L$ have been computed for $s \in \{1, \ldots, 10\}$ by finding hypothetical laser rays $\mathbf{d}_{sx}^L$ ($\mathbf{d}_{1l}^L$ is exemplified in red) and intersecting them with the checkerboard plane. Subsequently, the hypothetical laser rays are split into the sets $\mathbb{D}_{x_m}$. To indicate this distribution, we circle the corresponding border points. As can be seen, $\mathbf{d}_{5l}^L \notin \mathbb{D}_{l_1}$ and $\mathbf{d}_{5l}^L \notin \mathbb{D}_{l_2}$ since this hypothetical laser ray could belong to both sets, and is thus omitted.

Subsequently, the unit direction vectors $\mathbf{d}_i^L$ of the four boundaries, $i \in \{1, \ldots, 4\}$, can be computed as

$$\mathbf{d}_i^L = \frac{\mathbf{n}_{x_m}^L \times \mathbf{n}^L}{\|\mathbf{n}_{x_m}^L \times \mathbf{n}^L\|}, \qquad i = \begin{cases} 1, & \text{for } x = l, m = 1 \\ 2, & \text{for } x = l, m = 2 \\ 3, & \text{for } x = r, m = 1 \\ 4, & \text{for } x = r, m = 2 \end{cases} \tag{6.35}$$

$$\mathbf{d}_i^L \in [\mathbf{d}_i^L], \mathbf{n}_{x_m}^L \in [\mathbf{n}_{x_m}^L], \mathbf{n}^L \in [\mathbf{n}^L].$$

However, we are not able to determine all four direction vectors if the scan lines are more or less parallel to the boundaries of the checkerboard. In this case, we can only compute two of the four direction vectors.

Subsequently, it remains to associate which points belong to which line direction vectors. This association can be done straightforwardly by considering the sets $\mathbb{D}_{x_m}$. As can be seen from Fig. 6.8, each hypothetical laser ray $\mathbf{d}_{sx}^L \in \mathbb{D}_{x_m}$ has a corresponding border point $\mathbf{Q}_{sx}^L$. Since each hypothetical laser ray is already associated to one of the four checkerboard boundaries (according to the sets $\mathbb{D}_{x_m}$), we are able to unambiguously associate the corresponding border points as well. In the following, a border point that is associated to the boundary line i is denoted by $\mathbf{Q}_{ij}^L, j \in \{1, \ldots, N_i\}$, where N_i is the total number of points on the line i.

Finally, we exploit the fact that two of the four vectors each must be parallel due the geometry of the checkerboard. Thus, we intersect the corresponding pairs of direction vectors whose cross product domains contain 0 to reduce the uncertainty.

Point feature extraction

After finding the boundary lines, the corner points of the checkerboard can be extracted by performing a three-dimensional line intersection. Let $\mathbf{d}_i^L$ and $\mathbf{d}_j^L$ be two intersecting line direction vectors (i.e. their cross product must not be 0). Accordingly, $\mathbf{Q}_{ik}^L$ and $\mathbf{Q}_{jp}^L$ are points on the respective lines. This allows us to formulate the CSP $\mathcal{L}$:

$$\mathcal{L}: \begin{cases} \textbf{Variables: } \mathbf{C}_m^L, r_{ik}, r_{jp}, \mathbf{d}_i^L, \mathbf{d}_j^L, \mathbf{Q}_{ik}^L, \mathbf{Q}_{jp}^L \\ \textbf{Constraints:} \\ \quad 1. \quad \mathbf{C}_m^L = r_{ik}\mathbf{d}_i^L + \mathbf{Q}_{ik}^L = r_{jp}\mathbf{d}_j^L + \mathbf{Q}_{jp}^L \\ \textbf{Domains: } [\mathbf{C}_m^L], [r_{ik}], [r_{jp}], [\mathbf{d}_i^L], [\mathbf{d}_j^L], [\mathbf{Q}_{ik}^L], [\mathbf{Q}_{jp}^L] \end{cases} \tag{6.36}$$

$\mathbf{C}_m^L$ is the corner point we are interested in. To compute the domain $[\mathbf{C}_m^L]$, we have to contract the domains for r_{ik} and r_{jp}, which are unknown constants, and thus we set their initial domains to $[r_{ik}] = [r_{jp}] = [-\infty, \infty]$. To solve the CSP, we employ a forward-backward contractor and SIVIA. Since solely r_{ik} and r_{jp} are unknown, we have to bisect in two dimensions only.

To find the tightest possible domain for $\mathbf{C}_m^L$, we formulate the CSP $\mathcal{L}$ for every combination of points $\mathbf{Q}_{ik}^L$ and $\mathbf{Q}_{jp}^L$ (i.e. $\forall k \in \{1, \ldots, N_i\}$, $\forall p \in \{1, \ldots, N_j\}$) and intersect the resulting intervals for $\mathbf{C}_m^L$. Generally, we determine all four corner points (i.e. $m = \{1, \ldots, 4\}$) of the checkerboard. However, if we find only two parallel direction vectors, we cannot compute any corner point.

6.2.3 Finding the extrinsic parameters

Having extracted the desired features from both sensor data streams, we must now establish the correspondences between camera and laser features. Since there is only one plane on the checkerboard, finding the allocation between plane features is not necessary. However, as there are four boundaries and four corner points, the mapping of line and point features between both sensors is not as straightforward. For now, we assume this mapping to be known and will explain later how it can be established.

Before specifying the final CSP, we repeat the notions for all extracted features. Generally, a right superscript C or L indicates that the particular feature is given in the camera or laser scanner coordinate system, respectively.

- $\mathbf{n}^L$ and $\mathbf{n}^C$ are the unit checkerboard plane normal vectors.
- $\mathbf{d}_i^L$ and $\mathbf{d}_i^C$ are unit direction vectors describing the same checkerboard boundary line $i \in \{1, \dots, 4\}$.
- $\mathbf{Q}_{ij}^L$ and $\mathbf{Q}_{ik}^C$ are points on the line i with $j \in \{1, \dots, N_i\}$ and $k \in \{1, 2\}$. N_i is the total number of points on the line i which we extracted from laser scan data. In contrast, we determined only two points on every line i for the camera - namely the two adjacent corner points.
- $\mathbf{P}_l^L$ are scan points on the checkerboard with $l \in \{1, \dots, N_p\}$. N_p is the total number of scan points on the plane.
- d^C is the constant for the camera plane equation.
- $\mathbf{C}_m^L$ and $\mathbf{C}_m^C$ are corresponding checkerboard corner points.

Subsequently, we are able to formulate the CSP $\mathcal{C}$ which imposes constraints on the desired extrinsic calibration parameters composed of the rotation matrix $\mathbf{R}_L^C$ and the translation $\mathbf{T}_L^C$:

$$\mathcal{C}: \begin{cases} \textbf{Variables:} & \mathbf{R}_L^C, \mathbf{T}_L^C, \mathbf{n}^L, \mathbf{n}^C, \mathbf{d}_i^L, \mathbf{d}_i^C, \\ & \mathbf{Q}_{ij}^L, \mathbf{Q}_{ik}^C, \mathbf{P}_l^L, d^C, \mathbf{C}_m^L, \mathbf{C}_m^C \\ \textbf{Constraints:} & \\ & 1.\ \ \mathbf{R}_L^C \mathbf{n}^L = \mathbf{n}^C \\ & 2.\ \ \mathbf{R}_L^C \mathbf{d}_i^L = \mathbf{d}_i^C \\ & 3.\ \ \left(\mathbf{I} - \mathbf{d}_i^C \left(\mathbf{d}_i^C\right)^\intercal\right)\left(\mathbf{R}_L^C \mathbf{Q}_{ij}^L + \mathbf{T}_L^C - \mathbf{Q}_{ik}^C\right) = \mathbf{0} \\ & 4.\ \ \mathbf{n}^C \cdot \left(\mathbf{R}_L^C \mathbf{P}_l^L + \mathbf{T}_L^C\right) + d^C = 0 \\ & 5.\ \ \mathbf{R}_L^C \mathbf{C}_m^L + \mathbf{T}_L^C = \mathbf{C}_m^C \\ \textbf{Domains:} & [\mathbf{R}_L^C], [\mathbf{T}_L^C], [\mathbf{n}^L], [\mathbf{n}^C], [\mathbf{d}_i^L], [\mathbf{d}_i^C], \\ & [\mathbf{Q}_{ij}^L], [\mathbf{Q}_{ik}^C], [\mathbf{P}_l^L], [d^C], [\mathbf{C}_m^L], [\mathbf{C}_m^C] \end{cases} \tag{6.37}$$

The first two constraints establish a connection between unit direction vectors, and thus only involve the rotation matrix. The third constraint forces a boundary point $\mathbf{Q}_{ij}^L$, which is transformed into C, to also lie on the boundary as computed in C. The fourth constraint ensures that all scan points lying on the checkerboard fulfill the plane equation after being transformed into C. Lastly, the fifth constraint states that a corner point $\mathbf{C}_m^L$ which is transformed into C should coincide with the corresponding corner point computed from image data.

To reduce the number of unknown variables involved in the CSP $\mathcal{C}$ to six, we express the rotation matrix using Euler angles $\boldsymbol{\xi}_L^C$ (cf. Section 2.3.1). The domains $[\boldsymbol{\xi}_L^C]$ and $[\mathbf{T}_L^C]$ are initialized to either an initial estimate of the transformation (e.g. by examining the multi-sensor system) or to $[\boldsymbol{\xi}_L^C] = ([-\pi, \pi]_{\times 3})^\intercal$ and $[\mathbf{T}_L^C] = ([-\infty, \infty]_{\times 3})^\intercal$. In the second case, we assume no initial information about the extrinsic calibration. Subsequently, a forward-backward contractor is built for every constraint of the CSP $\mathcal{C}$. After combining these contractors to one large forward-backward contractor (cf. Section 3.7.2), we employ this contractor in combination with the SIVIA algorithm to find more accurate domains for the extrinsic transformation. In order to reduce computational load, we bisect only the domains for the Euler angles $[\boldsymbol{\xi}_L^C]$. Bisecting the domains for the three translation parameters is not necessary since they appear only once in each constraint, and are thus not as prone to overestimation as the Euler angles.

Up until here, we depicted the method to contract the extrinsic transformation parameters from one single laser scan and camera image. While this can already provide sufficiently precise estimates, we generally use data of different checkerboard poses to impose stronger constraints on the transformation parameters. Thus, we position the checkerboard in different poses in front of the camera and the laser scanner and gather multiple corresponding laser scans and camera images. Afterwards, we intersect all resulting forward-backward contractors and execute SIVIA as described above.

Naturally, different checkerboard poses provide varying constraints on each of the six transformation parameters. While one pose may particularly constrain the rotation around the z-axis, another pose may be especially valuable to contract the translation along the x-axis. Generally, we strive to contract the transformation parameter domains as much as possible while using as few checkerboard poses as needed, to limit the computational load. Thus, it is important to choose these checkerboard poses carefully. In this context, Zhou et al. [49] show that it is not necessary to move the checkerboard to yield all possible constraints. In contrast, they prove that rotating the checkerboard as close as possible to the sensors is sufficient.

Finding correspondences between camera and laser features

As suggested earlier, finding the correspondences between checkerboard boundaries or corner points is not a straightforward task. One possibility to find out which boundary (or corner point) as detected by the camera corresponds to which boundary (or corner point) as detected by the laser scanner, is to assume an initial estimate for the rotation matrix $\mathbf{R}_L^C \in [\mathbf{R}_L^C]$ and the translation vector $\mathbf{T}_L^C \in [\mathbf{T}_L^C]$. In this case, the domains $[\mathbf{R}_L^C]$ and $[\mathbf{T}_L^C]$ must be small enough to provide an unambiguous allocation between those features. The initial estimate may for example stem from the user who is able to observe the multi-sensor system.

However, generally we assume no initial information about the translation, i.e. $[\mathbf{T}_L^C] = ([-\infty, \infty]_{\times 3})^\intercal$, and the rotation, i.e. $[\boldsymbol{\xi}_L^C] = ([-\pi, \pi]_{\times 3})^\intercal$. To still find the correspondences, we can employ only the first and fourth constraints of the CSP $\mathcal{C}$, which do not require the mapping to be known. Subsequently, we are able to contract the extrinsic calibration domain by calling SIVIA in combination with a forward-backward contractor for this first and fourth constraint only. However, due the nature of Euler angles, an ambiguity may occur since different sets of Euler angles can describe the same rotation. Therefore, we must discard one of the possibilities or preclude such ambiguities by providing an initial estimate of the Euler angles.

Afterwards, it is possible to determine corresponding boundary lines using the second and third constraint of $\mathcal{C}$, and to finally find the corresponding corner points by employing the fifth constraint. In order to do that, we employ the contracted transformation domains to check for every possible mapping of features whether the corresponding constraint is fulfilled. If it is not, we can immediately conclude that the selected features do not belong to the same real world entity on the checkerboard. If a specific feature fulfills the constraints for multiple mappings, we cannot decide yet which mapping is correct. Solely in the case that a feature fulfills the constraints for only one corresponding feature, and vice versa, we can establish a mapping between these features (i.e. we require a bijection).

6.3 Conclusion

The contribution of this chapter are two new approaches for the guaranteed spatiotemporal calibration between different sensors. The first approach can be used to determine the time offset and the extrinsic rotation between camera and IMU. However, this approach includes the new time offset contractor $\mathcal{C}_{\text{offset}}$ that is more general and can be used to determine an interval enclosing the time offset between any two tubes. Therefore, this contractor constitutes a contribution to the field of constraint programming over dynamical systems. Moreover, this contractor can be applied to compute the time offset between any combination of sensors that measure the same entity over time.

To determine the extrinsic 6 DOF transformation between the coordinate systems of a camera and a laser scanner, this chapter additionally introduces a second approach. Here, the novelty of this work is that we present a method with which features of a checkerboard can be identified in a laser scanner point cloud in a guaranteed way. In combination with the guaranteed solution to the PnP problem in the previous chapter, this allows us to formulate a CSP for which we build forward-backward contractors that are capable of reliably contracting the transformation parameters. However, as for the PnP algorithm, we require bisections to compute the transformation between camera, laser scanner and IMU accurately. Consequently, in the future a dedicated contractor that does not decompose the constraints of the extrinsic calibration between sensors should be developed. Furthermore, the approach could be extended to other potentially three-dimensional calibration targets.

The experiments showing that we are able to reliably and accurately enclose the desired spatiotemporal calibration parameters using both simulated and real data can be found in Section 8.1 and Section 8.2.

To our knowledge, spatiotemporal calibration approaches between these sensor combinations have not yet been investigated in the context of bounded-error modeling, and thus pose an essential contribution of this work. Naturally, the spatiotemporal calibration parameters must be determined before information from camera, laser scanner and IMU can be fused. Since the aim of this work is to fuse sensor information in a guaranteed way to subsequently perform visual-LiDAR odometry, this chapter constitutes a necessary preliminary step in order to also compute the spatiotemporal calibration parameters in a guaranteed manner. In the following, we assume the intervals enclosing these parameters to be known, and can thus proceed with the main goal of this thesis: the interval-based sensor fusion for visual-LiDAR odometry.

7 Bounded-Error Visual-LiDAR Odometry

This chapter introduces our approach to perform dead reckoning for a mobile robot using data from camera, laser scanner and IMU. First, we fuse information from camera and laser scanner to create visual features that are augmented with depth information from the laser scanner. In order to do that, we assume both sensors to be synchronized (cf. Section 1.1.2). Moreover, in Section 6.2 we proposed an approach to compute the extrinsic transformation between camera and laser scanner under interval uncertainty. Using these extrinsic parameters and the bounded sensor error models introduced in Chapter 4, we are able to perform guaranteed visual-LiDAR sensor fusion, as all relevant errors are taken into account with intervals. Section 7.1 details our approach.

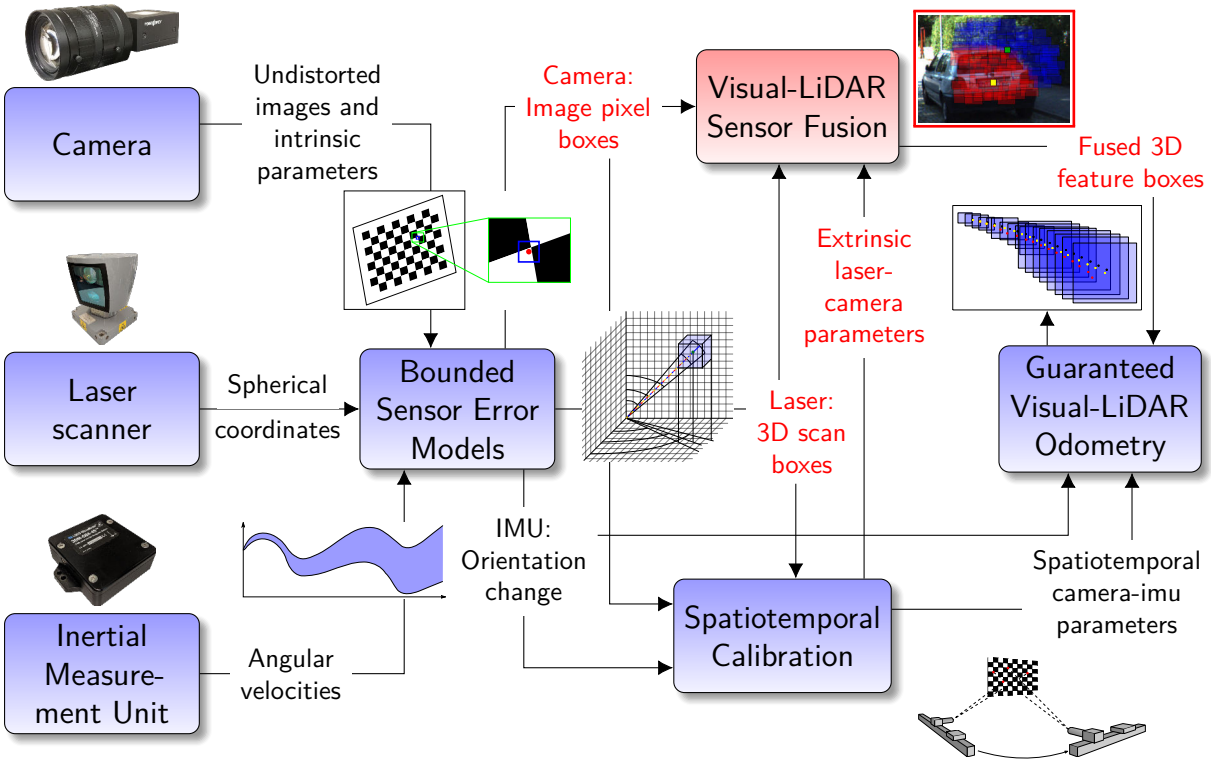


Figure 7.1: Classification of the sensor fusion in the overall context of this work.

Second, we employ the visual 3D features to estimate the robot's relative motion between consecutive points in time. In order to do that, the IMU data serves as an initial estimate for the rotation. Here, the spatiotemporal calibration parameters between camera and IMU

(cf. Section 6.1) are required to utilize this initial guess. Afterwards, we compute the rigid body transformation between two points in time that is constrained by the 3D features. Since our approach relies on interval analysis, and thus the computations are guaranteed, outliers can be unambiguously identified afterwards. Section 7.2 details our approach for the guaranteed visual-LiDAR odometry. Finally, Section 7.3 concludes this chapter by summarizing the procedure to incrementally localize a robot using camera, laser scanner and IMU.

7.1 Visual-LiDAR sensor fusion

The purpose of this section is to augment visual features with depth information for subsequent use in a more accurate motion estimation compared to visual features alone. In order to do that, camera and laser scanner information must first be brought into a common reference frame. Thus, in Section 7.1.1 we project the laser scan points onto the image plane. Afterwards, state-of-the-art methods are employed to detect and track image features between consecutive image frames. Finally, in Section 7.1.2 we present our approach to augment visual features with depth information in a guaranteed way.

7.1.1 Laser scan projection

To project the laser scan points onto the image plane, we have to first transform all points $i \in \{1, \ldots, N_l\}$ from the laser scanner coordinate system L into the camera coordinate system C, where N_l is the number of scan points. This can be done using the extrinsic transformation consisting of the rotation matrix $\mathbf{R}_L^C$ and the translation vector $\mathbf{T}_L^C$:

$$\mathbf{X}_i^C = \mathbf{R}_L^C \cdot \mathbf{X}_i^L + \mathbf{T}_L^C = \mathbf{R}_L^C \cdot \begin{pmatrix} r_i \sin\alpha_i \cos\beta_i \\ r_i \sin\alpha_i \sin\beta_i \\ r_i \sin\alpha_i \end{pmatrix} + \mathbf{T}_L^C, \tag{7.1}$$

where r_i, α_i and β_i are the spherical coordinates of laser scan point i according to Section 2.1.1.

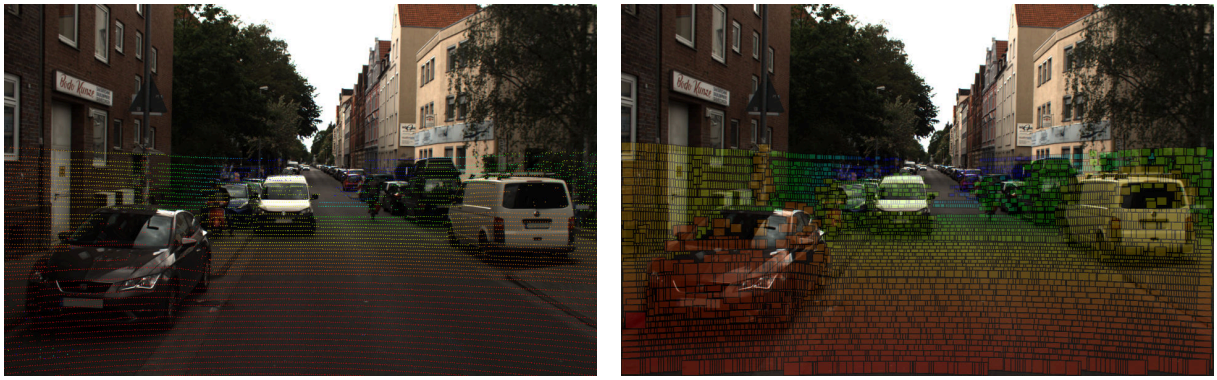
Subsequently, we project the transformed points onto the virtual image plane (z-coordinate equals 1) by normalizing their coordinates:

$$\begin{aligned} \tilde{x}_i^C &= \frac{x_i^C}{z_i^C}, \\ \tilde{y}_i^C &= \frac{y_i^C}{z_i^C}. \end{aligned} \tag{7.2}$$

Substituting (7.1) into (7.2) allows us to define the projection function

$$\begin{pmatrix} \tilde{x}_i^C \\ \tilde{y}_i^C \end{pmatrix} = \mathbf{f}_{\text{proj}}\left(\mathbf{X}_i^L, \mathbf{R}_L^C, \mathbf{T}_L^C\right) = \begin{pmatrix} \frac{\mathbf{R}_1 \cdot \mathbf{X}_i^L + T_1}{\mathbf{R}_3 \cdot \mathbf{X}_i^L + T_3} \\ \frac{\mathbf{R}_2 \cdot \mathbf{X}_i^L + T_2}{\mathbf{R}_3 \cdot \mathbf{X}_i^L + T_3} \end{pmatrix}, \tag{7.3}$$

where $\mathbf{R}_m$ and T_m, $m \in \{1, 2, 3\}$ are the m-th row of $\mathbf{R}_L^C$ and $\mathbf{T}_L^C$, respectively.



(a) Traditional projection without considering uncertainties.

(b) Interval boxes enclosing the projected laser scan points. The color of each box corresponds to the midpoint of the depth interval.

Figure 7.2: Exemplary projected laser scan points that are color coded by depth (red: close, blue: distant).

Since we perform computations with intervals, we define $[\mathbf{f}]_{\text{proj}}$ as the natural inclusion function of $\mathbf{f}_{\text{proj}}$ (cf. Section 3.6.1). Unfortunately, $[\mathbf{f}]_{\text{proj}}$ cannot be simplified. Thus, the coordinates $\mathbf{X}_i^L$ (and consequently the spherical coordinates r_i, α_i, β_i) and the entries of the rotation matrix (and consequently the Euler angles φ_L^C, θ_L^C, ψ_L^C) appear several times. To avoid substantial overestimation of the projected scan points, it is possible to bisect the spherical coordinates and the Euler angles during the computation [9].

Finally, we compute the interval box containing the projected laser scan point as

$$[\tilde{\mathbf{X}}_i^C] = \begin{pmatrix} [\tilde{x}_i^C] \\ [\tilde{y}_i^C] \end{pmatrix} = [\mathbf{f}]_{\text{proj}} \left([\mathbf{X}_i^L], [\mathbf{R}_L^C], [\mathbf{T}_L^C] \right), \tag{7.4}$$

where $[\mathbf{X}_i^L]$ is the box enclosing the scan point i according to the bounded error model derived in Section 4.2.1. Moreover, $[\mathbf{R}_L^C]$ and $[\mathbf{T}_L^C]$ are the domains enclosing the extrinsic calibration parameters that can be determined as depicted in Section 6.2.

Fig. 7.2 shows laser scan points that are projected onto the image plane in the traditional way (i.e. without considering uncertainties) and using the approach we proposed. The depth interval of a projected scan point i is its z-coordinate before the projection, i.e. $[z_i^C]$.

7.1.2 Data fusion

Before we can fuse data from camera and laser scanner, we have to decide which image points we want to augment with depth information. Naturally, any approach could be employed to detect and match image features. Since this is not the focus of our work, we decided to use one that is available in Open Source Computer Vision Library (OpenCV) [114] and has produced reasonable results in our experiments. Consequently, we employ "Good features to track" which is a state-of-the-art approach to detect features in the image [115]. Subsequently, we use the Lucas Kanade feature tracker to calculate the optical flow between consecutive images, and are thus able to re-identify the aforementioned image features [116]. In this way, we are able to establish so-called image feature matches, meaning that the matched image features of the

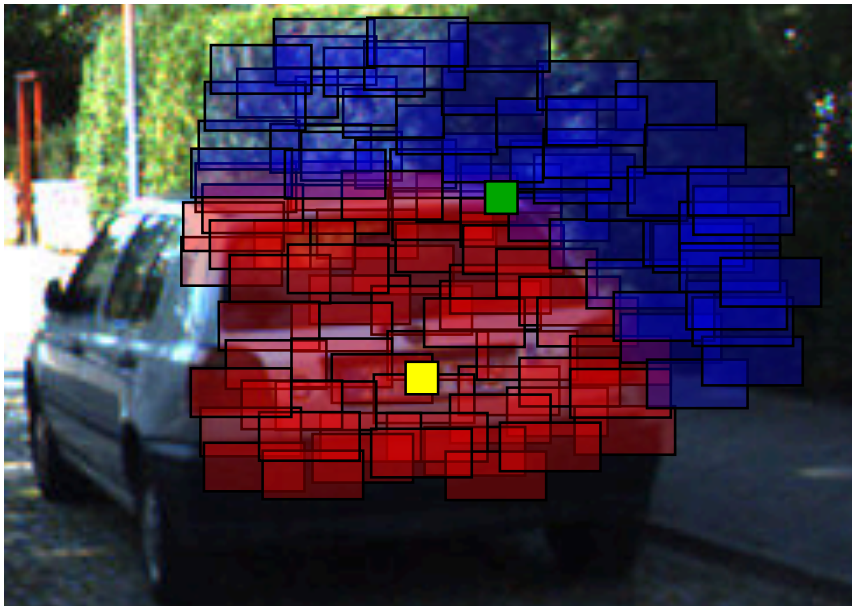


Figure 7.3: Our general idea to fuse data from camera and laser scanner. For this example, we only distinguish between scan boxes that are close (red) or distant (blue). As can be seen, the yellow image feature resides on a plane that is almost parallel to the image plane, and thus all intersecting scan boxes exhibit similar depth information. In contrast, the green image feature resides on an edge and intersects scan boxes from both fore- and background. Consequently, the depth estimate for the yellow image feature will be more accurate than for the green image feature.

first and second image correspond to the same object. Naturally, outliers can occur during this feature matching process. However, for now we assume the established matches to be correct and take care of outliers in Section 7.2.4.

Fig. 7.3 shows the general idea of our approach to fuse data from camera and laser scanner. Since image features and scan points are no longer points, but boxes guaranteed to contain the true value, we find all overlapping scan boxes for each image feature box. Consequently, we compute the depth of the image feature as the union over the depth of all intersecting scan boxes. This results in an interval for the depth which allows us to immediately assess its accuracy. In the following, we formalize our method.

Let j be an image feature which we aim to augment by depth information. First, we compute the box enclosing its normalized coordinates on the image plane according to the bounded error model introduced in Section 4.2.2: $[\tilde{\mathbf{X}}_j^C] = [\tilde{x}_j^C] \times [\tilde{y}_j^C]$.

Next, we determine the set $\mathcal{S}_j$ of all previously projected scan boxes i whose normalized coordinates have a non-empty intersection with the normalized coordinates of feature j:

$$\mathcal{S}_j = \{\, i \in \{1, \ldots, N_l\} \mid [\tilde{\mathbf{X}}_i^C] \cap [\tilde{\mathbf{X}}_j^C] \neq \varnothing \,\}. \tag{7.5}$$

Subsequently, we can determine the depth (i.e. the z-coordinate) of the image feature j. In order to do that, we compute the union over the z-coordinates of all overlapping scan boxes:

$$[z_j^C] = \bigcup_{i \in \mathcal{S}_j} [z_i^C]. \tag{7.6}$$

Consequently, we use this interval to determine the non-normalized x- and y-coordinates:

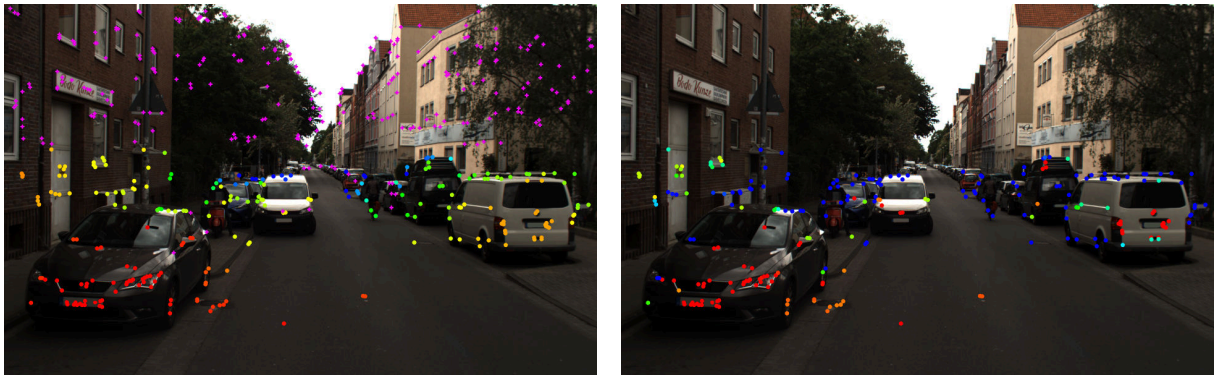
$$[\mathbf{X}_j^C] = [z_j^C] \begin{pmatrix} [\tilde{x}_j^C] \\ [\tilde{y}_j^C] \\ 1 \end{pmatrix}, \tag{7.7}$$

where $[\mathbf{X}_j^C]$ is the desired box enclosing the 3D coordinates of the image feature j.

Of course, the underlying assumption to find depth information using our proposed method is that the laser scanner measures a 3D point close enough to every image feature. Otherwise, it might happen that the true depth of a feature is not enclosed in the computed interval. However, this is the case since the laser scanner produces a dense point cloud and we check that each image feature is surrounded by sufficient 3D points.

The result of this approach are depth augmented visual features that can be re-identified in successive images. Since we compute the depth estimates using interval analysis, the results are guaranteed if the initial assumptions about the sensor errors are correct. Moreover, we can directly assess the accuracy of the depth estimates, and are thus able to distinguish between accurate and inaccurate features. For example, the depth of features residing on a plane parallel to the image plane can generally be computed more accurately than for features residing on edges. Fig. 7.4 shows exemplary images that illustrate the results of our sensor fusion approach.

However, we cannot calculate the depth of every feature, for example, if the projected point cloud does not cover the entire image. Nevertheless, image features without depth information can still be employed for the motion estimation. Consequently, in the following section we distinguish between features with or without depth information.



(a) Colored by depth (red: close, blue: distant). Features without depth information are colored purple.

(b) Colored by depth accuracy (red: accurate, blue: inaccurate).

Figure 7.4: Exemplary results of our approach for data fusion. In the left image, image features are color coded by their depth, i.e. the mid point of the interval enclosing their z-coordinate. In the right image, image features are colored by their depth accuracy, i.e. the width of the interval enclosing their z-coordinate. As expected, the depth estimate is less accurate for features that are close to edges since the feature could lie on either side of the edge (e.g. features on the borders of cars). Furthermore, the depth estimate for features that are far away from the camera is also as expected less accurate.

7.2 Guaranteed visual-LiDAR odometry

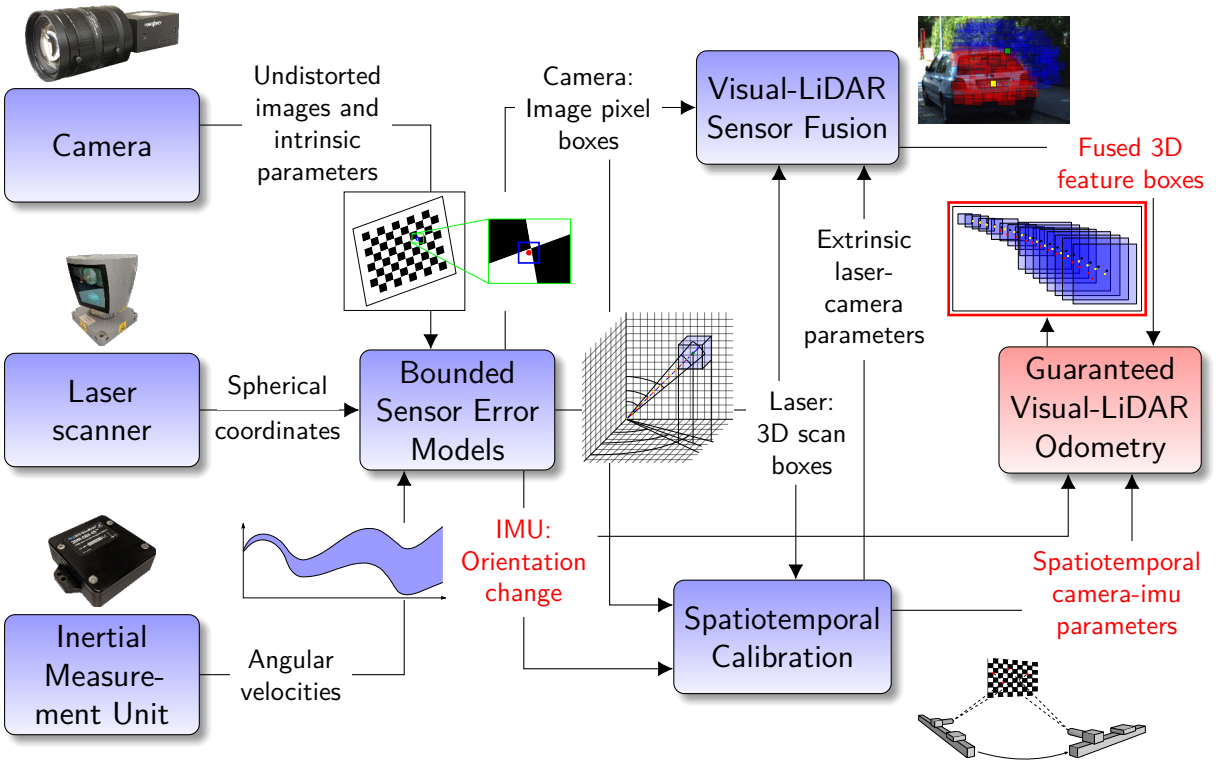


Figure 7.5: Classification of the guaranteed visual-LiDAR odometry in the overall context of this work.

In this section we introduce our approach to incrementally localize a robot using the previously fused information from camera and laser scanner as well as IMU data. As explained in Section 2.6, no global information (e.g. a known map or known pose in a global coordinate system) is available. Thus, we can only compute the relative motion of the robot between distinct points in time, which are the times of acquiring image and laser scan data for our work. Fig. 7.6 shows the general idea of our approach.

Formally, we define the problem of visual-LiDAR odometry as follows. Let f and g be two consecutive image frames with the corresponding timestamps t_f^C and t_g^C such that $t_g^C > t_g^C$. The motion of the robot between these image frames is described by the rigid body transformation consisting of the rotation matrix $\mathbf{R}_{C_g}^{C_f}$ and the translation vector $\mathbf{T}_{C_g}^{C_f}$. Here, C_f and C_g is the camera coordinate system during the acquisition of image frames f and g, respectively.

First, Section 7.2.1 introduces the procedure to compute an initial enclosure of the robot's relative rotational movement between image frames. Here, the spatiotemporal calibration parameters between camera and IMU that are determined according to Section 6.1 are required. Next, in Section 7.2.2 we explain how to employ the previously created visual 3D features to further contract the initial rotation estimate and simultaneously compute the relative translational movement of the robot. Moreover, although it is possible to compute the relative motion between each consecutive image pair, we introduce the concept of keyframes in Section 7.2.3 to reduce the drift of our approach. Last, in Section 7.2.4 we explain how

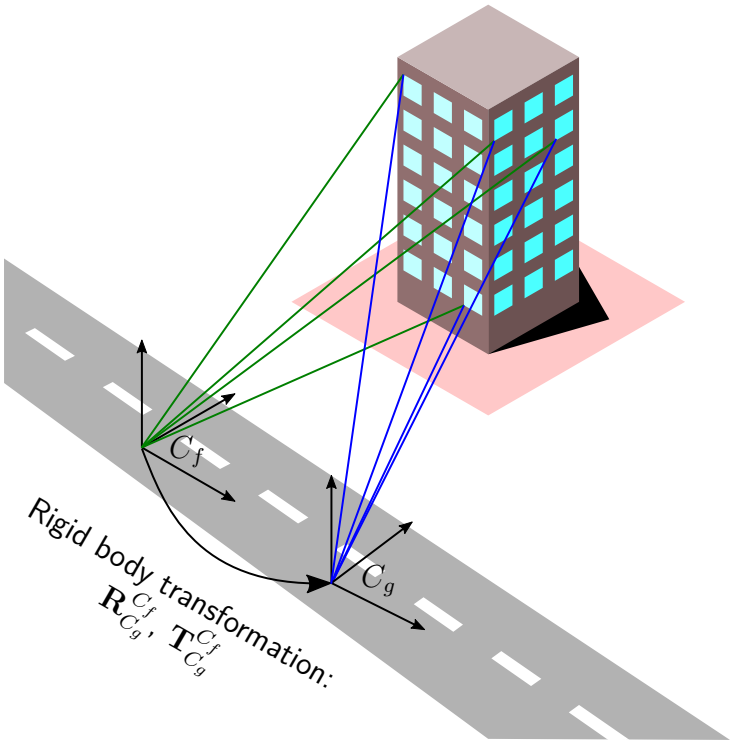


Figure 7.6: We find visual features and augment them by depth information in two different coordinate systems (i.e. at two different poses), and are thus able to find the rigid body transformation consisting of the rotation matrix $\mathbf{R}_{C_g}^{C_f}$ and the translation vector $\mathbf{T}_{C_g}^{C_f}$.

outliers occurring during the image feature matching process can be taken into account during the motion estimation.

7.2.1 Rotation prediction using IMU data

The goal of this section is to compute intervals that serve as a first approximation of the relative rotation of our robot between the image frames f and g. Again, t_f^C and t_g^C are the points in time at which the camera captures the images f and g, respectively. Generally, given these two points in time, we can use the unknown but bounded angular velocities measured by the IMU and integrate them according to equation (2.12). However, there exists an unknown but bounded time offset $[\tau]$ between the data streams of the camera and the IMU that we determine as explained in Section 6.1. Consequently, in addition to the uncertainty of the angular velocities, we also have to take into account the uncertainty of this time offset. Here, the general idea is to translate the time uncertainty into an increased uncertainty of the angular velocities.

Let $[\hat{\boldsymbol{\omega}}^I](\cdot)$ be the three-dimensional tube enclosing the angular velocities in the time reference and coordinate system of the IMU that is built according to the error model in Section 4.2.3. Now, the goal is to determine the three-dimensional tube $[\boldsymbol{\omega}^I](\cdot)$ that encloses

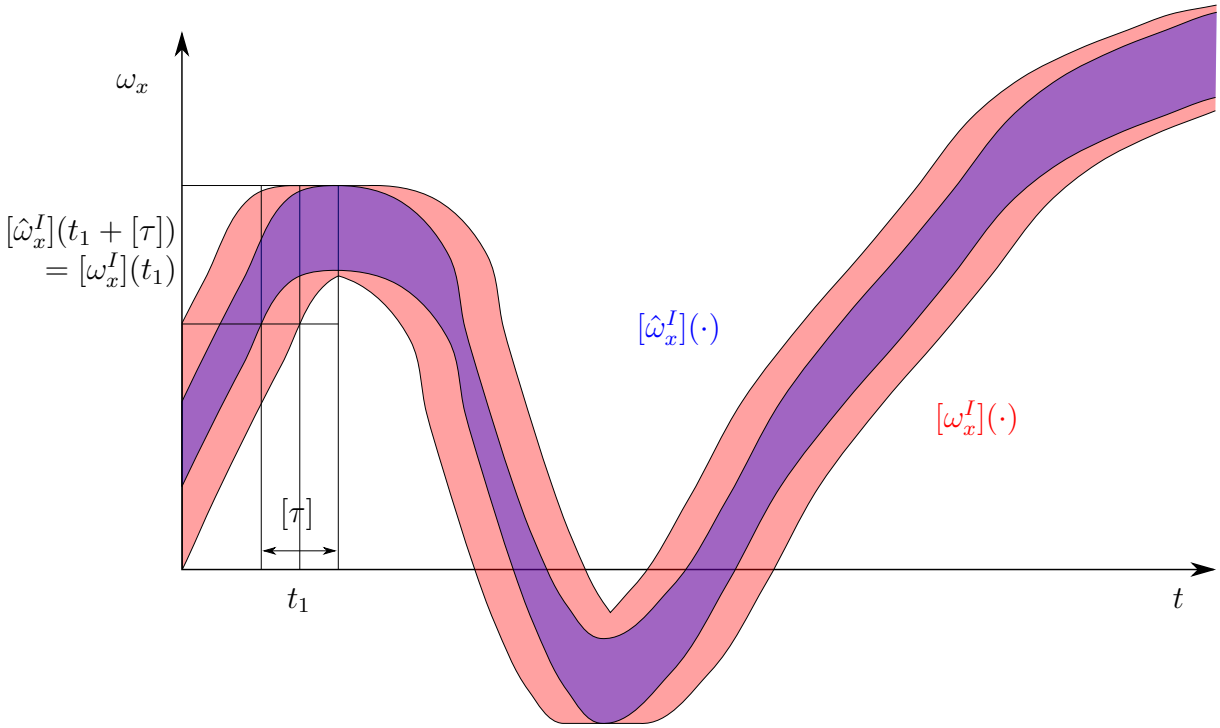


Figure 7.7: Visualization of how the uncertainty of the angular velocities is inflated to account for the unknown but bounded time offset $[\tau]$.

the angular velocities in the time reference of the camera. Consequently, we have to take the time offset $[\tau]$ into account and compute the desired tube as:

$$\forall t \in [t_0, t_f] : [\boldsymbol{\omega}^I](t) = [\hat{\boldsymbol{\omega}}^I](t + [\tau]), \tag{7.8}$$

where $[t_0, t_f]$ is the time interval over which $[\hat{\boldsymbol{\omega}}^I](\cdot)$ is defined. Fig. 7.7 visualizes this idea. Note, however, that the angular velocities still correspond to the coordinate system of the IMU and we only account for the time offset between camera and IMU, but not yet for the extrinsic rotation between both sensors.

Subsequently, we employ the natural inclusion function of (2.12) to integrate the angular velocities $[\boldsymbol{\omega}^I](\cdot)$ between t_f^C and t_g^C. This results in an interval matrix $[\mathbf{R}_{I_g}^{I_f}]$ that encloses the relative rotation of the IMU between the acquisition of the image frames f and g by the camera.

Remark

In theory, since (2.12) *corresponds to a simple first-order integration method, the interval matrix* $[\mathbf{R}_{I_g}^{I_f}]$ *is not guaranteed to enclose the true rotation matrix* $\mathbf{R}_{I_g}^{I_f\,*}$*. This is due to the assumption of angular velocities being constant for a small period of time.*

If we want to keep the guarantees intact, it is possible to use guaranteed integration methods [117] and formulate the problem as follows:

$$\dot{\mathbf{x}}_t = \mathbf{f}\left(\mathbf{x}_t, \hat{\boldsymbol{\omega}}^I(t)\right), \tag{7.9}$$

where $\mathbf{f}$ *is the differential equation corresponding to the matrix formulation in* (2.9) *and*

$$\begin{aligned} \mathbf{x}_t &= \begin{pmatrix} r_{11} & r_{12} & \dots & r_{33} & \Upsilon \end{pmatrix}, \\ \mathbf{x}_0 &= \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \tau \end{pmatrix}, \\ \dot{\Upsilon} &= 1, \\ \mathbf{x}_t \in [\mathbf{x}_t],\ \mathbf{x}_0 \in [\mathbf{x}_0],\ \tau \in [\tau],\ \hat{\boldsymbol{\omega}}^I(t) &\in [\hat{\boldsymbol{\omega}}^I](t), \end{aligned} \tag{7.10}$$

where $\mathbf{x}_0$ is the initial condition and $r_{11}, r_{12}, \dots, r_{33}$ are the desired entries of the rotation matrix. Note that the time offset $[\tau]$ is taken into account by introducing the parameter Υ in the differential equation that evolves with time due to the additional constraint $\dot{\Upsilon} = 1$ and setting the initial condition $\Upsilon_0 \in [\tau]$.

However, the guaranteed integration of (7.9) *is not straightforward and up until now there exists no software library to compute it. Consequently, this work relies on the first-order integration method described above. Although this approach is theoretically not guaranteed, it has no disadvantages in practice because the angular velocities are measured at a high frequency.*

Finally, after computing the robot's rotation in the IMU coordinate system, we have to transform it into the camera coordinate system. In order to do that, we require the extrinsic rotation between both sensor coordinate systems that can be determined as explained in Section 6.1. Consequently, we compute

$$[\mathbf{R}_{C_g}^{C_f}] = \left([\mathbf{R}_C^I]\right)^\mathsf{T} \cdot [\mathbf{R}_{I_g}^{I_f}] \cdot [\mathbf{R}_C^I], \tag{7.11}$$

where $[\mathbf{R}_C^I]$ is the interval matrix enclosing the extrinsic rotation matrix.

7.2.2 Rigid body transformation

From here, we omit the superscript C for the rotation matrix $\mathbf{R}_{C_g}^{C_f}$ and the translation vector $\mathbf{T}_{C_g}^{C_f}$. Instead, we write $\mathbf{R}_g^f$ and $\mathbf{T}_g^f$ since all computations are performed in the camera coordinate system. Moreover, we also omit the superscript C for coordinates and write, for example, $\mathbf{X}_j^f$ instead of $\mathbf{X}_j^{C_f}$ for the coordinates of a feature j in the camera coordinate system C during the image frame f. To reduce the number of unknowns for the rotation matrix $\mathbf{R}_g^f$, we express it using the Euler angles $\boldsymbol{\xi}_g^f = (\, \varphi_g^f \;\; \theta_g^f \;\; \psi_g^f \,)$ (cf. Section 2.3.1).

After computing an initial enclosure for the robot's relative rotation between the image frames f and g, this section introduces constraints to further contract the rotation estimate $\mathbf{R}_g^f$ and to compute the translational movement $\mathbf{T}_g^f$. In order to do that, we distinguish between three cases for matched image features. First, we build a contractor for image features which were successfully augmented by depth information in both images. Second, we employ the contractors introduced for the PnP problem (cf. Chapter 5) for image features for which depth information was found only in one of the two consecutive images. Third, we build a new contractor for image features that could not be augmented by depth information in both images.

7.2.2.1 Depth information in both images

If we succeed to augment an image feature by depth information in both images, we know its 3D coordinates which allows us to formulate the rigid body transformation as follows:

$$\mathbf{X}_j^f = \mathbf{R}_g^f \cdot \mathbf{X}_j^g + \mathbf{T}_g^f, \tag{7.12}$$

where $\mathbf{X}_j^f$ are the 3D coordinates of the feature $j \in \mathcal{F}_{\text{both}}$ found in image f and $\mathbf{X}_j^g$ is the same feature j found in image g. Moreover, $\mathcal{F}_{\text{both}}$ is the set of all matches for which both features were augmented by depth information in image g and f.

This allows us to formulate a multi-dimensional function $\mathbf{f}^{\text{both}}$ with the j-th row as

$$\mathbf{f}_j^{\text{both}}\left(\boldsymbol{\xi}_g^f, \mathbf{T}_g^f, \mathbf{X}_j^f, \mathbf{X}_j^g\right) = \mathbf{R}_g^f\left(\boldsymbol{\xi}_g^f\right) \cdot \mathbf{X}_j^g + \mathbf{T}_g^f - \mathbf{X}_j^f = \mathbf{0}. \tag{7.13}$$

Now, we can formulate the CSP $\mathcal{H}_{\text{both}}$:

$$\mathcal{H}_{\text{both}} : \begin{pmatrix} \forall j \in \mathcal{F}_{\text{both}} : \quad \mathbf{f}_j^{\text{both}}\left(\boldsymbol{\xi}_g^f, \mathbf{T}_g^f, \mathbf{X}_j^f, \mathbf{X}_j^g\right) = \mathbf{0} \\ \boldsymbol{\xi}_g^f \in [\boldsymbol{\xi}_g^f], \mathbf{T}_g^f \in [\mathbf{T}_g^f], \mathbf{X}_j^f \in [\mathbf{X}_j^f], \mathbf{X}_j^g \in [\mathbf{X}_j^g] \end{pmatrix}, \tag{7.14}$$

where $[\boldsymbol{\xi}_g^f]$ and $[\mathbf{T}_g^f]$ are the desired interval vector enclosing the Euler angles and the translation parameters, respectively. Furthermore, $[\mathbf{X}_j^f]$ and $[\mathbf{X}_j^g]$ are the 3D coordinates of the feature j in the images f and g, respectively, that are augmented by depth information as explained in Section 7.1.2.

Subsequently, we build a forward-backward contractor $\mathcal{C}_j^{\text{both}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f])$ (cf. Section 3.7.2) for each constraint j of the CSP $\mathcal{H}_{\text{both}}$ and intersect all of them to obtain the final contractor

$$\mathcal{C}_{\text{both}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]) = \bigcap_{j \in \mathcal{F}_{\text{both}}} \mathcal{C}_j^{\text{both}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]). \tag{7.15}$$

7.2.2.2 Depth information in one image

If we fail to find depth information for a feature in image f, but succeed to do so for the same feature in image g, the problem of estimating the rotation matrix and the translation vector becomes the PnP problem:

$$z_j^f \tilde{\mathbf{X}}_j^f = \mathbf{R}_g^f \cdot \mathbf{X}_j^g + \mathbf{T}_g^f, \tag{7.16}$$

where z_j^f is the unknown depth and $\tilde{\mathbf{X}}_j^f$ are the normalized image coordinates of a feature $j \in \mathcal{F}_{\text{one}}$. Moreover, $\mathcal{F}_{\text{one}}$ is the set of all matches for which only one feature was augmented by depth information.

Vice versa, if we fail to find depth information for a feature in image g, but succeed to do so for the same feature in image f, the equation becomes

$$\left(\mathbf{R}_g^f\right)^\intercal \left(\tilde{\mathbf{X}}_j^f - \mathbf{T}_g^f\right) = z_j^g \mathbf{X}_j^g. \tag{7.17}$$

In both cases we can employ the contractors introduced in Section 5.1. However, as there are generally only a few features for which depth information is found in one image but not in

the other, we omit the linear Gauss-Seidel contractor since it requires at least six such features. Thus, we only build the nonlinear forward-backward contractor $\mathcal{C}_j^{\mathsf{pnp,nl}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f])$ for every feature $j \in \mathcal{F}_{\mathsf{one}}$.

Again, we intersect all these contractors to obtain the final contractor $\mathcal{C}_{\mathsf{one}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f])$:

$$\mathcal{C}_{\mathsf{one}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]) = \bigcap_{j \in \mathcal{F}_{\mathsf{one}}} \mathcal{C}_j^{\mathsf{pnp,nl}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]). \tag{7.18}$$

7.2.2.3 No depth information

If no depth information is available for a feature in both images, the equation becomes:

$$z_j^f \tilde{\mathbf{X}}_j^f = z_j^g \mathbf{R}_g^f \cdot \tilde{\mathbf{X}}_j^g + \mathbf{T}_g^f, \tag{7.19}$$

where z_j^f and z_j^g are the unknown depths of the image feature in image f and g, respectively. To eliminate both z_j^f and z_j^g we combine all three rows. First, we solve the third row for z_j^f:

$$z_j^f = z_j^g (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) + T_3, \tag{7.20}$$

where R_l, $l \in \{1,2,3\}$ is the l-th row of $\mathbf{R}_g^f$ and T_m, $m \in \{1,2,3\}$ is the m-th row of $\mathbf{T}_g^f$. Next, we substitute the previous expression (7.20) for z_j^f into the second row of (7.19):

$$(z_j^g (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) + T_3)\tilde{y}_j^f = z_j^g (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g) + T_2. \tag{7.21}$$

Rearranging the expression allows us to find an expression for z_j^g:

$$\begin{aligned} z_j^g \tilde{y}_j^f (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) - z_j^g (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g) &= T_2 - \tilde{y}_j^f T_3 \\ z_j^g &= \frac{T_2 - \tilde{y}_j^f T_3}{\tilde{y}_j^f (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) - (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g)}. \end{aligned} \tag{7.22}$$

Finally, both (7.20) and (7.22) can be substituted into the first row of (7.19):

$$\tilde{x}_j^f \left(\frac{T_2 - \tilde{y}_j^f T_3}{\tilde{y}_j^f (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) - (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g)} (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) + T_3 \right) = \frac{T_2 - \tilde{y}_j^f T_3}{\tilde{y}_j^f (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) - (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g)} (\mathbf{R}_1 \cdot \tilde{\mathbf{X}}_j^g) + T_1. \tag{7.23}$$

Eliminating the common denominator on both sides results in

$$\begin{aligned} \tilde{x}_j^f \left(T_2 - \tilde{y}_j^f T_3\right) (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) + \tilde{x}_j^f T_3 \left(\tilde{y}_j^f (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) - (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g)\right) = \\ \left(T_2 - \tilde{y}_j^f T_3\right) (\mathbf{R}_1 \cdot \tilde{\mathbf{X}}_j^g) + T_1 \left(\tilde{y}_j^f (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) - (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g)\right). \end{aligned} \tag{7.24}$$

Rearranging this expression results in

$$\left(-\tilde{y}_j^f T_3 + T_2\right) (\mathbf{R}_1 \cdot \tilde{\mathbf{X}}_j^g) + \left(\tilde{x}_j^f T_3 - T_1\right) (\mathbf{R}_2 \cdot \tilde{\mathbf{X}}_j^g) + \left(-\tilde{x}_j^f T_2 + \tilde{y}_j^f T_1\right) (\mathbf{R}_3 \cdot \tilde{\mathbf{X}}_j^g) = 0, \tag{7.25}$$

which can be expressed in matrix form as

$$\begin{pmatrix} -\tilde{y}_j^f T_3 + T_2 & \tilde{x}_j^f T_3 - T_1 & -\tilde{x}_j^f T_2 + \tilde{y}_j^f T_1 \end{pmatrix} \cdot \mathbf{R}_g^f \cdot \tilde{\mathbf{X}}_j^g = 0. \tag{7.26}$$

We define $j \in \mathcal{F}_{\text{no}}$, where $\mathcal{F}_{\text{no}}$ is the set of all image feature matches without depth information. Consequently, we obtain a multi-dimensional function $\mathbf{f}^{\text{no}}$ with the j-th row as

$$\begin{aligned} &\mathbf{f}_j^{\text{no}}\left(\boldsymbol{\xi}_g^f, \mathbf{T}_g^f, \tilde{\mathbf{X}}_j^f, \tilde{\mathbf{X}}_j^g\right) = \\ &\begin{pmatrix} -\tilde{y}_j^f T_3 + T_2 & \tilde{x}_j^f T_3 - T_1 & -\tilde{x}_j^f T_2 + \tilde{y}_j^f T_1 \end{pmatrix} \cdot \mathbf{R}_g^f\left(\boldsymbol{\xi}_g^f\right) \cdot \tilde{\mathbf{X}}_j^g = 0, \end{aligned} \tag{7.27}$$

Now, we can formulate the CSP $\mathcal{H}_{\text{no}}$:

$$\mathcal{H}_{\text{no}} : \begin{pmatrix} \forall j \in \mathcal{F}_{\text{no}} : & \mathbf{f}_j^{\text{no}}\left(\boldsymbol{\xi}_g^f, \mathbf{T}_g^f, \tilde{\mathbf{X}}_j^f, \tilde{\mathbf{X}}_j^g\right) = 0 \\ \boldsymbol{\xi}_g^f \in [\boldsymbol{\xi}_g^f], \mathbf{T}_g^f \in [\mathbf{T}_g^f], & \tilde{\mathbf{X}}_j^f \in [\tilde{\mathbf{X}}_j^f], \tilde{\mathbf{X}}_j^g \in [\tilde{\mathbf{X}}_j^g] \end{pmatrix}, \tag{7.28}$$

where $[\boldsymbol{\xi}_g^f]$ and $[\mathbf{T}_g^f]$ are the desired interval vector enclosing the Euler angles and the translation parameters, respectively. Furthermore, $[\tilde{\mathbf{X}}_j^f]$ and $[\tilde{\mathbf{X}}_j^g]$ are the normalized image coordinates of the feature j in the images f and g, respectively.

Subsequently, we build a forward-backward contractor $\mathcal{C}_j^{\text{no}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f])$ (cf. Section 3.7.2) for each constraint j of the CSP $\mathcal{H}_{\text{no}}$ and intersect all of them to obtain the final contractor

$$\mathcal{C}_{\text{no}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]) = \bigcap_{j \in \mathcal{F}_{\text{no}}} \mathcal{C}_j^{\text{no}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]). \tag{7.29}$$

7.2.2.4 Final contractor

After introducing the contractors in the previous sections, we can now build one contractor that combines all of them to provide a more accurate enclosure of the robot's motion. Since we distinguish between image features for which depth information can be found in either both image frames, one image frame or not at all, we build three main contractors. Each of these contractors is an intersection over multiple forward-backward contractors that are built for each individual image feature match. Consequently, we can intersect the three main contractors to build one common contractor that successively applies the primitive forward-backward contractors:

$$\mathcal{C}_{\text{odom}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]) = \mathcal{C}_{\text{both}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]) \ \cap \ \mathcal{C}_{\text{one}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]) \ \cap \ \mathcal{C}_{\text{no}}([\boldsymbol{\xi}_g^f], [\mathbf{T}_g^f]). \tag{7.30}$$

We refrain from employing more sophisticated contractors than forward-backward contractors since these exhibit several advantages for our application. First, they can deal with highly nonlinear constraints which arise in our application due to the three-dimensional rotations [9, 73]. Second, in contrast to other contractors, the number of constraints can be larger than the number of unknowns [9, 73]. In general, this is the case since there are only six unknowns (6 DOF transformation), but we detect significantly more image feature matches. Third, the computational cost for forward-backward contractors is comparatively low, as it is only necessary to decompose the constraint into primitive functions, compute the forward evaluation

and then propagate the new domains backwards [9, 73]. This is important in view of the fact that we employ constraints from a large number of image features.

However, the main drawback of the forward-backward contractor is that variables that appear more often in the equation cannot be contracted optimally due to the dependency problem (cf. Section 3.6.1). This is the case for the Euler angles in our constraints. Thus, we additionally employ bisections to compute these parameters more accurately, as explained in Section 7.3.

7.2.3 Keyframes

Since the concept of *keyframes* has been proven to be successful in many approaches for visual odometry, we adopt it in this work [57]. This means that instead of computing the robot's relative motion between consecutive image frames, we estimate the transformation from the current image frame relative to the most recently defined keyframe until we have to insert a new keyframe. Fig. 7.8 shows the idea.

The main advantage of this concept is that the odometry drift can reduced since the error accumulates only between keyframes and not between each consecutive image frame. For example, when computing the motion frame by frame, we would have to determine the rotation between k_1 and k_2 as follows:

$$\mathbf{R}_{k_2}^{k_1} = \mathbf{R}_{f_1}^{k_1} \cdot \mathbf{R}_{f_2}^{f_1} \cdot \cdots \cdot \mathbf{R}_{k_2}^{f_7}. \tag{7.31}$$

As can be easily understood, this leads to an accumulation of the error over all frames. In contrast, finding image feature matches between k_1 and k_2 allows the direct computation of the rotation $\mathbf{R}_{k_2}^{k_1}$, which is thus independent of the intermediate frames.

However, this raises the question of when to insert a new keyframe. On the one hand, it is beneficial to insert as few keyframes as possible to reduce the computation time and minimize drift. On the other hand, it is important to track sufficient reliable visual features for a robust motion estimation. Established approaches employ simple (number and distribution of image features) or more advanced heuristics to judge the quality of their visual features. However, these metrics are often not resilient to different circumstances as becomes evident by the fact

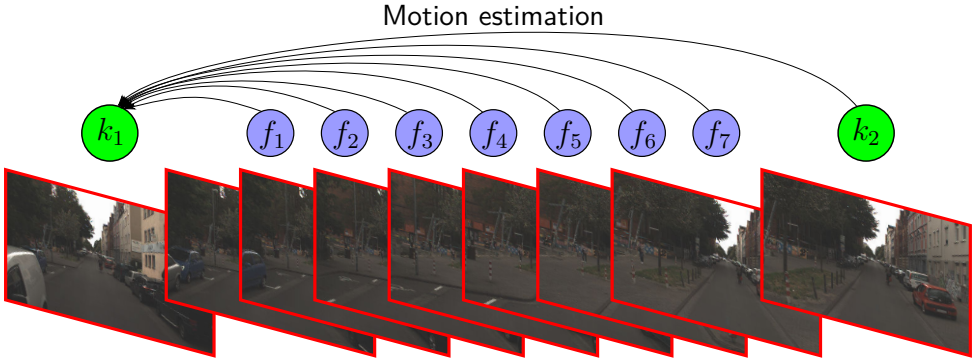


Figure 7.8: Visualization of the concept of keyframes. Instead of estimating the motion between consecutive images (i.e. $\mathbf{R}_{f_2}^{f_1}$, $\mathbf{R}_{f_3}^{f_2}$, etc.), we estimate the motion relative to the last keyframe k_1 (i.e. $\mathbf{R}_{f_1}^{k_1}$, $\mathbf{R}_{f_2}^{k_1}$, etc.) until we have to insert a new keyframe k_2.

that different approaches for the selection of keyframes have been introduced [57, 65, 118, 119].

In contrast, in the context of interval analysis, we can employ a straightforward approach to select keyframes. We propose to use a dynamic approach that observes the uncertainty of odometry estimates (i.e. the width of the intervals for the Euler angles and the translation parameters) and inserts a new keyframe once the position uncertainty exceeds a predefined threshold. An increasing uncertainty is the direct consequence of insufficient constraints that prevent our contractors from contracting the pose domain, and thus a direct indicator of insufficient visual features. This allows us to insert keyframes only when it is absolutely necessary. Moreover, this approach can be straightforwardly adapted for different applications by setting the maximum tolerable uncertainty.

7.2.4 Outlier treatment

As explained in Section 7.1.2, the 3D features needed to estimate the rigid body transformation are not necessarily outlier-free. Although our approach to fuse data from camera and laser scanner is guaranteed to compute a proper enclosure for the depth of a visual feature, the feature matching between images is error-prone. Accordingly, it may happen that the feature tracking algorithm matches two visual features that do not correspond to the same object. Moreover, we may detect image features that correspond to the same object, but this object is not static (e.g. a moving car), and thus must be treated as outliers when computing the rigid body transformation.

To deal with outliers occurring during the feature matching, we employ a two-fold approach. First, we aim to exclude outliers before computing the rigid body transformation. Second, we account for remaining outliers during the motion estimation by employing a relaxed intersection (cf. Section 3.9).

We exclude wrongly matched image features before proceeding with the motion estimation using different approaches. First, we employ a state-of-the-art approach to make the computation of the optical flow of image features more robust. The basic idea is that the solution should not depend on the reference image, i.e. it should not matter whether the optical flow is calculated from image A to image B, or vice versa [120]. Thus, we only keep image feature matches that are established independent of the direction in which the optical flow is computed.

Second, given the corresponding image features, we compute the so-called fundamental matrix using Random Sample Consensus (RANSAC). This matrix relates the image points from image A to another image B, i.e. using the fundamental matrix, for each image point from image A we can describe a so-called epipolar line in image B on which the corresponding point must lie. Consequently, we mark all matched image features that deviate by more than the maximum image pixel error $r([\Delta_{px}])$ (cf. Section 4.2.2) from these epipolar lines as outliers. More details on the background of epipolar geometry can be found in [28].

Third, we use the 3D information of depth augmented image features if it is available for a feature match in both image frames. The main idea is that for a rigid body, the distance between two points does not change over time. Thus, for every two 3D features i and j, the distances measured in two consecutive frames g and f must coincide. Let us denote the 3D

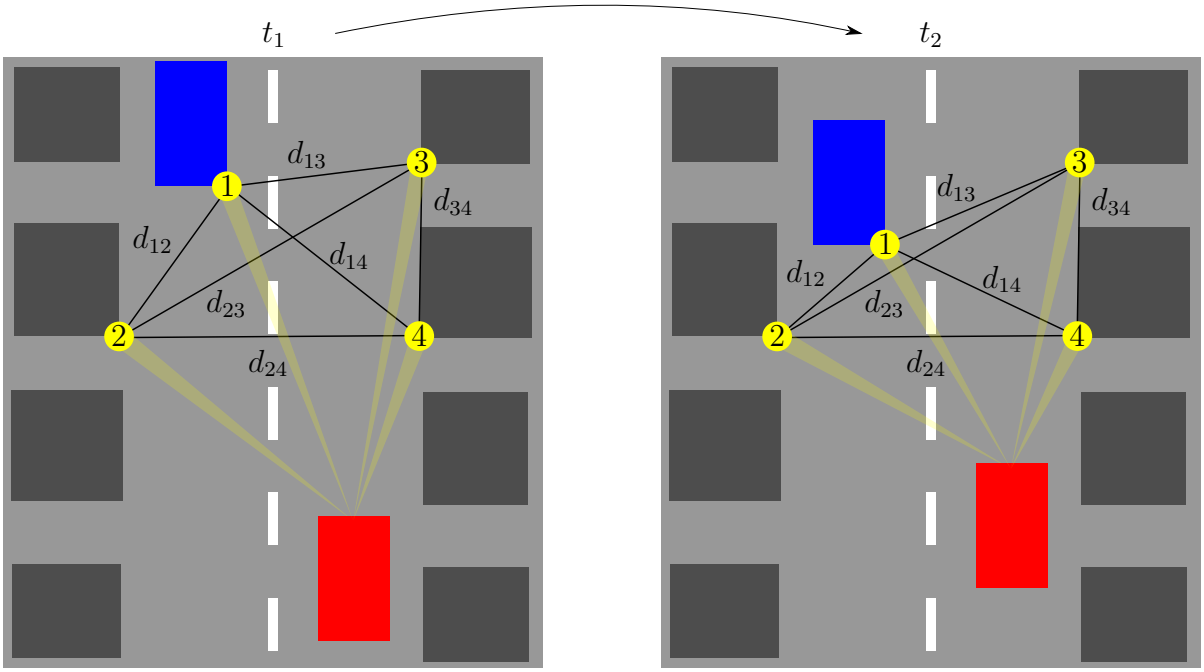


Figure 7.9: Visualization of the idea to detect outliers by computing the distances between 3D points. At t_1, the red robot detects four yellow features and computes their 3D coordinates as explained above. Subsequently, we compute the distances between these 3D points. After moving to t_2, we repeat the same procedure and check whether the distances remain the same. However, since the first feature resides on a moving object, the distances $d_{1j}, j \in \{2, 3, 4\}$ change. Consequently, we are able to mark the first feature as an outlier. Similarly, we can detect outliers occurring due to mismatched features.

coordinates of a point i in frame g by $\mathbf{X}_i^g$ and the corresponding Euclidean distance between the points i and j in frame g by d_{ij}^g.

As introduced before, we do not have the exact values of the 3D coordinates, but only intervals of possible values. Based on these intervals, for each i and j, we can only compute intervals of possible values of d_{ij}^g and d_{ij}^f. The actual value of the distance must be in both intervals, thus these intervals must have a non-zero intersection. To compute the intervals for the distance, we use the natural inclusion function of the Euclidean distance (cf. Section 3.6.1).

If at least one of the points is mismatched or not rigid (i.e. an outlier for our application) and the corresponding intervals are tight enough, the distances become different, and there is no longer any intersection. Consequently, we aim to find the largest set of points that are mutually consistent (i.e. the distances between any two points of the set must intersect over both frames). Since the problem of finding this set of points is NP-complete, we employ the approach proposed by Howard [121]:

1. We determine the point that is consistent with the largest number of other points and add it to the set.
2. For each point that is not in the set, we compute the number of points from the set that it is consistent with.
3. We add the point with the largest number computed in step two to the set.

Consequently, we repeat steps two and three until no more points can be added to the set. Finally, only the 3D points from the set are employed for the rigid body transformation, while all other points are marked as outliers. Fig. 7.9 visualizes the idea.

Afterwards, we use the Guaranteed Minimum Outlier Number Estimator (GOMNE) [122]. The idea of this algorithm is to initially assume all remaining 3D features to be correct and use the forward-backward contractors (cf. Section 7.2.2.4) to compute a solution set. If this solution set is empty, the number of constraints is iteratively relaxed using a q-relaxed intersection (i.e. q is increased in every iteration) until a non-empty solution set is found.

While this procedure helps to remove obvious outliers, it is not guaranteed to find all outliers. Thus, some may remain and distort the motion estimation. Therefore, we employ a q-relaxed intersection (cf. Section 3.9) when computing the final forward-backward contractor $\mathcal{C}_{\text{odom}}$ in Section 7.2.2.4. In order to do that, we have to specify the maximum number of outliers to expect. However, this is a difficult assessment to make since this number depends on the environment and visible dynamic objects. Consequently, we tend to choose the maximum number of outliers more conservatively, which means that we rather over- than underestimate this number. Nevertheless, we can determine it only empirically during our experiments.

Finally, after computing the rigid body transformation using the relaxed intersection, we aim to identify the remaining outliers. Interval analysis allows us to do so in a guaranteed way. Given the contracted transformation parameters, we check if the constraints formulated in Section 7.2.2 hold for the corresponding image feature matches. If this is not the case, we have identified an outlier in a guaranteed way. However, it is not possible to guarantee that all outliers will be found. This is because the depth information of a feature might be computed inaccurately (e.g. for features on edges). Consequently, it would be consistent with any transformation although the image feature match is incorrect.

7.3 Summary of the visual-LiDAR odometry approach

To conclude this chapter, we give an overview over our algorithm. Fig. 7.10 shows a flowchart that details the sequence of the previously introduced steps. As can be seen, we couple the resulting forward-backward contractor with the SIVIA algorithm to further contract the initial rotation estimate $[\boldsymbol{\xi}_f^k]$ and compute intervals enclosing the translational movement $[\mathbf{T}_f^k]$. Since only the Euler angles appear multiple times in the constraints, and are thus subject to the dependency problem, we only bisect these three parameters.

Naturally, the result of this computation is only the transformation between the robot's pose at the last keyframe and its current pose. Thus, we have to consider the robot's motion up until the keyframe to compute its pose relative to the start. Fig. 7.11 shows an exemplary transformation chain with keyframes and regular frames in between. Consequently, the robot's pose at frame f_9 relative to the first keyframe k_1 can be computed as

$$\mathbf{R}_{f_9}^{k_1} = \mathbf{R}_{k_2}^{k_1} \cdot \mathbf{R}_{k_3}^{k_2} \cdot \mathbf{R}_{f_9}^{k_3}, \tag{7.32}$$

$$\mathbf{T}_{f_9}^{k_1} = \mathbf{T}_{k_2}^{k_1} + \mathbf{R}_{k_2}^{k_1} \cdot \mathbf{T}_{k_3}^{k_2} + \mathbf{R}_{k_2}^{k_1} \cdot \mathbf{R}_{k_3}^{k_2} \cdot \mathbf{T}_{f_9}^{k_3}, \tag{7.33}$$

with $\mathbf{R}_b^a \in [\mathbf{R}_b^a]$, $\mathbf{T}_b^a \in [\mathbf{T}_b^a]$, $a \in \{k_1, k_2, k_3\}$, $b \in \{k_2, k_3, f_9\}$.

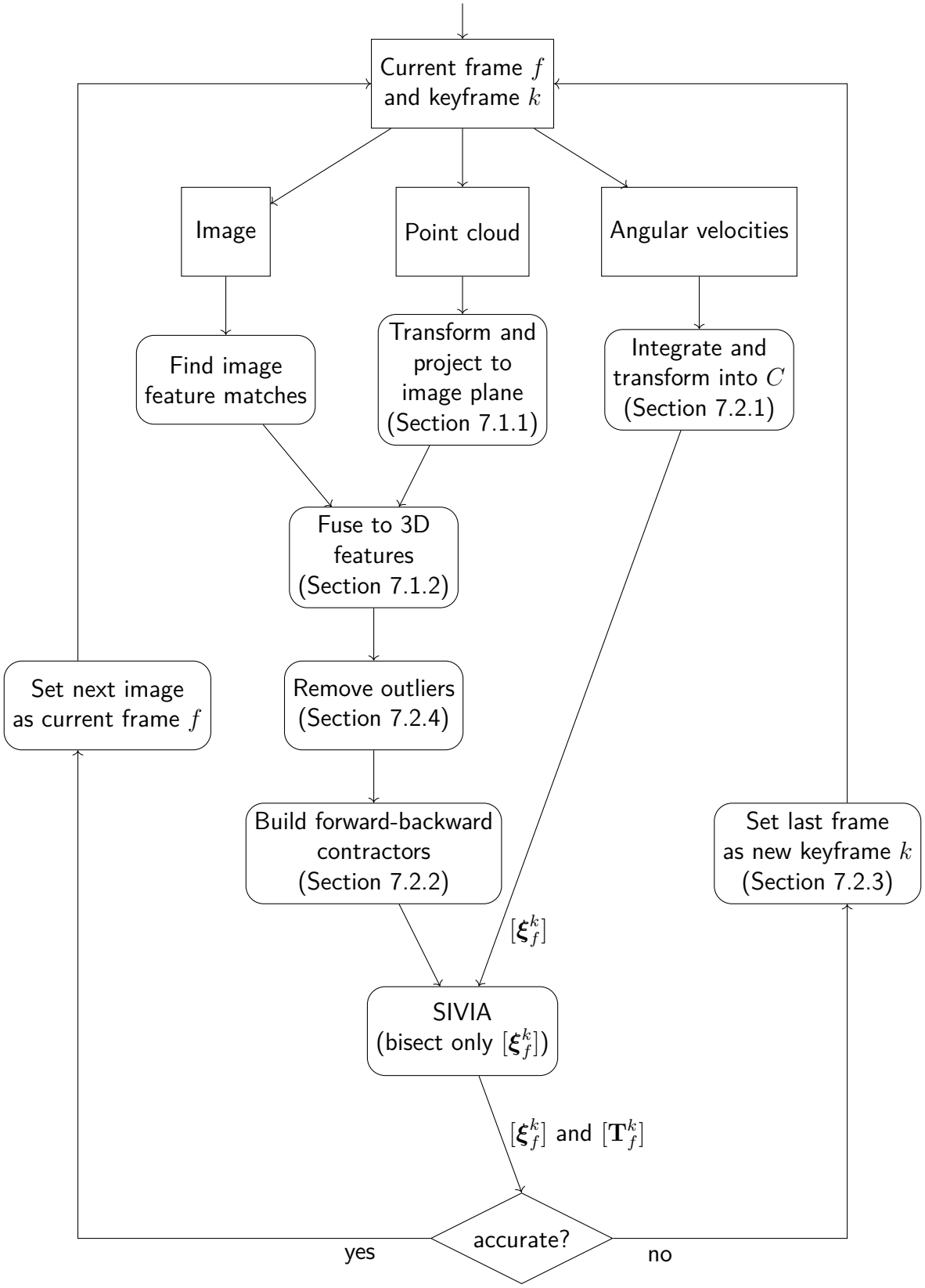


Figure 7.10: Overview of our approach for bounded-error visual-LiDAR odometry. The same procedure is repeated for every new image that is acquired at the same time as a corresponding point cloud from the laser scanner.

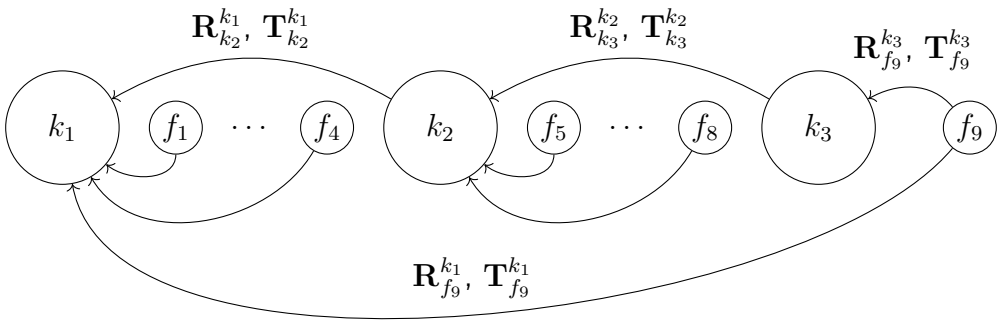


Figure 7.11: Exemplary transformation after the insertion of three keyframes.

7.4 Conclusion

This chapter concludes the methodology section of this work by introducing a novel approach for guaranteed visual-LiDAR odometry. Here, the contributions lie in the consideration of spatiotemporal uncertainties, the original method for sensor data fusion, and the new strategy for keyframe insertion. Accounting for the uncertainty of the extrinsic transformation between sensor coordinate systems can be done by simply transforming information between those coordinate systems using the interval transformation parameters. To consider a time offset between camera and IMU, we propagate the time uncertainty to the uncertainty of the measured angular velocities before integrating them. Here, a guaranteed integration method could be employed in the future that we deemed unnecessary for this work due to the high measuring frequency of the IMU.

Since all sensor errors are modeled rigorously, we are able to generate visual 3D features that can be re-identified over time and carry information about their depth uncertainty. As a result, features whose depth information is uncertain contribute weaker constraints to the CSP, and thus have less influence on the pose estimation. Again, we build forward-backward contractors for the CSP and couple them with SIVIA to accurately estimate the 6 DOF transformation. As mentioned earlier, these contractors are not ideal since they decompose the problem, and thus introduce pessimism. Consequently, contractors dedicated to the problem of 6 DOF pose estimation should be developed in the future.

Our novel strategy for keyframe insertion is based on the direct link between the uncertainty of the pose estimates and the quality of the visual 3D features. This is due to the fact that less accurate features lead to a less accurate contraction of the robot's pose. Consequently, we insert a new keyframe as soon as the uncertainty of the estimated pose exceeds a predefined threshold.

Section 8.3 depicts an evaluation of the guaranteed visual-LiDAR odometry using real data from a large-scale experiment. Since our dataset also contains precise ground truth information, we can show that our approach is indeed able to enclose the robot's pose in a guaranteed way.

8 Experimental Results

In this chapter, the newly introduced error models and approaches are evaluated and investigated with regard to their applicability for guaranteed sensor fusion. Moreover, we compare our approaches to established stochastic approaches to highlight advantages and disadvantages.

The experiments are carried out both in simulated, but also, more importantly, in real environments. First, simulated data helps to understand the general accuracy of our approaches and allows us to vary sensor error bounds or specific parameters to study their effects. Second, real data confirms the applicability of our error models and approaches to real multi-sensor systems and mobile robots.

All computations are performed on a single consumer-grade laptop (Intel Core i7 @ 2.7 GHz $\times$ 8, 16 GB RAM). To acquire and store measurement data we use the Robot Operating System (ROS) [123]. In addition, we employ the IBEX library [69] for constraint processing over real numbers, the Tubex library [78] for computations over sets of trajectories and VIBes [124] to visualize interval boxes.

8.1 Spatiotemporal calibration between camera and IMU

This section details the experimental evaluation of the approach for the interval-based spatiotemporal calibration between camera and IMU that we presented in Section 6.1. The main evaluation criterion is the desired guarantee of being able to reliably enclose the true solution. Since it is difficult to find the true solution using real data, we first evaluate the approach using simulated data before showing that it is also capable of dealing with real data.

The results of this experiment have previously been presented in [27]. Therefore, we reuse some of the wording of this publication in this section.

8.1.1 Simulated data

To be able to know the true time offset and extrinsic rotation between camera and IMU, we first use simulated data. Moreover, this enables us to simulate different offsets which should be enclosed by our approach. As a prerequisite for the experiments carried out, the experimental environment used is first explained.

8.1.1.1 Simulated experimental environment

To simulate the experiment, we employed the multi-robot simulator *Gazebo* [125]. It is convenient to use with the Robot Operating System (ROS) as necessary interfaces are available by default. Furthermore, common sensors for autonomous robots such as laser scanners, cameras and IMUs, which we require for our experiments, are readily available.

First, we constructed a simulation environment that is composed of the calibration target and the multi-sensor system consisting of camera and laser scanner. In addition, we added a motor to the multi-sensor system which allows to rotate the setup around all three axis in front of the calibration target. To compute the orientation of the camera with respect to the calibration target more accurately, we decided to assemble our calibration target from three checkerboards, which are mounted perpendicular to each other. Fig. 8.1 shows the calibration target. The size of each board is 34×34 cm, while the imprinted checkerboard is composed of 6×6 squares with a side length of 5 cm each. To distinguish the three different checkerboards, they are augmented by Aruco markers [126]. We placed the calibration target in a distance of roughly 2 m from the multi-sensor system.

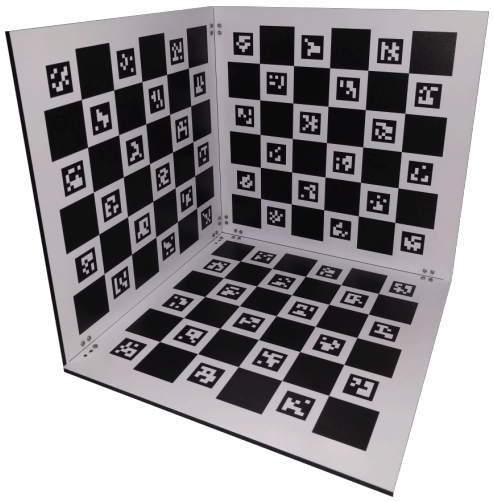


Figure 8.1: Image of the calibration target for the spatiotemporal calibration between camera and IMU.

We used Gazebo's predefined camera controller to simulate the camera. Unfortunately, it only allows to simulate Gaussian noise for each frame and does not provide an option to define the error distribution manually. We simulated an error following a Gaussian distribution with a mean value of 0 and a standard deviation of 0.01. This noise value was added to each pixel's color channels which lie in the range from 0 to 1. Furthermore, the controller allows to manually set the parameters of the pinhole camera model (cf. Section 2.1.2). We set similar parameters as we encounter for the real camera - i.e. we set an image resolution of 1920×1080 px and $f_x = f_y = 2200$ px, $c_x = 960$ px, $c_y = 540$ px. A subsequent camera calibration showed a maximum reprojection error of 0.3 px, and thus we set $[\Delta_{px}] = [-0.3, 0.3]$ px. Moreover, we set the frame rate of the camera to 25 frames per second (FPS).

Likewise, we used Gazebo's predefined IMU controller to simulate gyroscope measurement. As for the camera, we replicated the IMU that we use for the real experiments. Consequently, we introduced the different types of error mentioned in Section 4.2.3 for the simulation of the angular

velocities, namely the scale factor Δ_s, the constant bias $\mathbf{\Delta}_b$, the walking bias $\mathbf{\Delta}_{wb}$ and the measurement noise $\mathbf{\Delta}_n$. The values for these parameters are set to comply with the data sheet of our real IMU. Consequently, we set the same error bounds that we later derive in Section 8.1.2.2 for the experiments with real data: $[\Delta_s] = [-0.05, 0.05]\,\%$, $[\mathbf{\Delta}_b] = ([-0.05, 0.05]\,^\circ/\mathrm{s})^\intercal_{\times 3}$, $[\mathbf{\Delta}_{wb}] = ([-0.0014, 0.0014]\,^\circ/\mathrm{s}^2)^\intercal_{\times 3}$ and $[\mathbf{\Delta}_{wb}] = ([-0.03, 0.03]\,^\circ/\mathrm{s})^\intercal_{\times 3}$.

The true orientation of the camera in the IMU frame is specified using MRP (cf. Section 2.3.2) as $\boldsymbol{\rho}_C^I = (-\frac{1}{3} \;\; -\frac{1}{3} \;\; -\frac{1}{3})^\intercal$ for all simulations. We chose these rotation parameters to approximately replicate our real multi-sensor system.

8.1.1.2 Results

We recorded a set of four different data sets that vary by the axis around which the multi-sensor system is rotated and by the velocity with which the setup is rotated in front of the calibration target. The setup was rotated around one axis only for the first data set (e.g. only around the axis facing upwards). Our experiments show that the results are independent of the choice of this axis. For the three remaining data sets the setup was rotated around all three axis at the same time. The motion durations for all data sets range from 1 s to 3 s.

Time offset only

At first, we solve the one-dimensional problem, which means that we only use the time offset contractor $\mathcal{C}_{\text{offset}}$ to contract $[\tau]$ and assume the extrinsic rotation between camera and IMU to be known up to some accuracy. The results for two different rotation uncertainties $[\boldsymbol{\rho}_C^I]$ are depicted in Table 8.1. The column containing $[\tau_1]$ corresponds to a rather small rotation uncertainty $[\boldsymbol{\rho}_C^I] = ([-0.34, -0.32]_{\times 3})^\intercal$, which results in an angle uncertainty of $w([\theta_C^I]) = 6°$ according to (2.19). In contrast, we choose a large rotation uncertainty $[\boldsymbol{\rho}_C^I] = ([-0.36, -0.30]_{\times 3})^\intercal$ for the column containing $[\tau_2]$. This corresponds to an angle uncertainty of $w([\theta_C^I]) = 18°$.

Axis	Velocity (rad/s)	$[\tau_1]$ (ms)	$[\tau_2]$ (ms)
one axis	2.0	$[-11.7, 13.3]$	$[-14.8, 14.1]$
three axis	0.5	$[-37.5, 50.0]$	$[-39.8, 100.0]$
three axis	1.0	$[-20.3, 28.1]$	$[-24.2, 62.5]$
three axis	2.0	$[-10.9, 17.2]$	$[-14.8, 39.1]$

Table 8.1: Results of $\mathcal{C}_{\text{offset}}$ to only compute the time offset $[\tau]$ for different simulated experiments that vary by the rotation axis and the rotational velocity.

It can be seen that the true offset $\tau^* = 0$ is enclosed within $[\tau_1]$ and $[\tau_2]$ for all four data sets. Furthermore, we note that the velocity has a large influence on the resulting interval width. Another important finding is that the offset interval widths increase for $[\tau_2]$, but not by the same factor we increase the rotation uncertainty. The rotational velocity has a far greater impact. This shows that we can find a reasonable timestamp offset interval even if we are uncertain about the rotation between both sensors. The smallest interval we obtain is

$[\tau] = [-11.7, 13.3]$ ms, and thus the highest accuracy we achieve using only the time offset contractor $\mathcal{C}_{\text{offset}}$ is $w([\tau]) = 25$ ms.

Full spatiotemporal calibration

Next, we solve the four-dimensional problem which means that we also assume the extrinsic rotation between camera and IMU to be unknown, and thus have to bisect the intervals for the MRP $[\boldsymbol{\rho}_C^I]$ and simultaneously use $\mathcal{C}_{\text{offset}}$ to contract the interval enclosing the time offset $[\tau]$. Here, we set an accuracy of $\epsilon = 0.001$ for the SIVIA algorithm.

Moreover, we choose the initial rotation $[\boldsymbol{\rho}_C^I] = ([-0.5, -0.1]_{\times 3})^\intercal$, which corresponds to an initial angle uncertainty of $w([\theta]) = 124°$. Generally, an initial rotation estimate is not needed if the multi-sensor system is rotated around all three axis to accurately enclose the true extrinsic rotation. However, since we also recorded data in which we rotated the setup around one axis only, an initial, albeit uncertain, rotation estimate is required to provide a solution in this case as well.

Axis	Velocity (rad/s)	$[\tau]$ (ms)	$[\rho_x]$	$[\rho_y]$	$[\rho_z]$	$w([\theta])$ (°)
one axis	2.0	$[-10.2, 11.7]$	$[-0.42, -0.20]$	$[-0.50, -0.10]$	$[-0.50, -0.10]$	102.79
three axis	0.5	$[-33.6, 36.7]$	$[-0.37, -0.29]$	$[-0.37, -0.30]$	$[-0.36, -0.31]$	19.98
three axis	1.0	$[-16.4, 19.5]$	$[-0.37, -0.29]$	$[-0.37, -0.30]$	$[-0.36, -0.31]$	20.06
three axis	2.0	$[-8.6, 10.9]$	$[-0.37, -0.29]$	$[-0.37, -0.30]$	$[-0.36, -0.31]$	19.91

Table 8.2: Results of the full spatiotemporal calibration for different simulated experiments that vary by the rotation axis and the rotational velocity. The single components of the MRP are denoted as follows: $[\boldsymbol{\rho}_C^I] = (\,[\rho_x]\;\;[\rho_y]\;\;[\rho_z]\,)^\intercal$.

Table 8.2 shows the results. As expected, we are not able to find a tight enclosure for the extrinsic rotation if we rotate around one axis only. However, if we rotate around all three axis, we can reduce the initial interval widths to find an outer approximation for the rotation parameters. As can be seen, the accuracy of the rotation parameters does not depend on the rotational velocity. Furthermore, we are able to increase the accuracy of the timestamp offset by solving the four-dimensional problem. The smallest interval we obtain is $[\tau] = [-8.6, 10.9]$ ms, which corresponds to an uncertainty of $w([\tau]) = 19.5$ ms. Nevertheless, this comes at the cost of an increased computation time in the region of several minutes up to one hour while the one-dimensional problem can be solved in less than 0.1 s.

Different true time offsets

Next, we present results for different simulated time offsets between camera and IMU. For this experiment, we only use the fourth data set of the previous evaluation in which we rotated our setup around all three axis with a velocity of 2 rad/s. We show results for both the one-dimensional and the four-dimensional problem. For the one-dimensional problem

we only employ the time offset contractor $\mathcal{C}_{\text{offset}}$ and set $[\boldsymbol{\rho}_C^I] = ([-0.34, -0.32]_{\times 3})^\intercal$, which corresponds to an angle uncertainty of $w([\theta_C^I]) = 6°$. For the four-dimensional problem we additionally bisect the intervals for the extrinsic rotation between both sensors starting from an initial domain of $[\boldsymbol{\rho}_C^I] = ([-0.5, -0.1]_{\times 3})^\intercal$, which corresponds to an initial angle uncertainty of $w([\theta]) = 124°$.

τ^*	$[\tau_{1D}]$ (ms)	$[\tau_{4D}]$ (ms)
10.0	$[-0.8, 27.3]$	$[0.8, 21.1]$
-10.0	$[-21.1, 7.8]$	$[-18.8, 1.6]$
20.0	$[8.6, 37.5]$	$[10.9, 31.3]$
50.0	$[39.1, 67.2]$	$[41.4, 60.9]$

Table 8.3: Results of $\mathcal{C}_{\text{offset}}$ and the full spatiotemporal calibration for different simulated time offsets.

The results can be seen in Table 8.3. $[\tau_{1D}]$ corresponds to the one-dimensional problem while $[\tau_{4D}]$ corresponds to the four-dimensional problem. Note, that the true offset τ^* is always enclosed in the corresponding intervals $[\tau_{1D}]$ and $[\tau_{4D}]$. If we solve the four-dimensional problem, the extrinsic rotation between the sensors is also recovered with an accuracy of approximately $w([\theta]) = 20°$. Furthermore, it can be seen that the deviation of the true offset from zero has no effect on the accuracy of the computed offset. Small fluctuations can be explained by the nature of the discretization into tube slices and the bisections in Algorithm 5.

8.1.2 Real data

Since our approach is able to reliably enclose the true time offset and extrinsic rotation between camera, we can evaluate its applicability to real data. Because of the previously mentioned problem of finding ground truth information for our real multi-sensor system, this section only evaluates whether a reasonably accurate solution can also be found for real data, and compares the results to the outcome of an established approach. As a prerequisite for the experiments carried out, the multi-sensor system and the experimental environment is introduced.

8.1.2.1 Real experimental environment

Fig. 8.2 shows the multi-sensor system consisting of a camera and an IMU that we employed to record data for the experimental evaluation of the spatiotemporal calibration. It is composed of a FLIR Grasshopper3 GS3-U3-23S6C-C color camera and a LORD MicroStrain 3DM-GQ4-45 GNSS aided IMU. The camera is equipped with a Fujinon CF12.5HA-1 lens that has a focal length of 12.5 mm.

For our experiments we operated the camera at a resolution of 1920×1200 px and a frame rate of 25 FPS. Before acquiring data, we calibrated our camera intrinsically using the algorithm proposed by Zhang [17] and the corresponding implementation in OpenCV [114]. The focal length amounts to approximately 2200 px. Using the intrinsic calibration parameters we removed distortion from each image before processing it. The coordinate system of the

Figure 8.2: Image of the multi-sensor system consisting of a camera and an IMU.

camera is defined as follows: the z-axis is pointing in the viewing direction, the x-axis is pointing to the right and the y-axis is pointing down. Besides, the IMU is operated at a frequency of 100 Hz.

Moreover, we employed the calibration target that was already introduced in the simulated experimental environment and is depicted in Fig. 8.1. Since we cannot precisely access the printing accuracy of the checkerboard, we conservatively assume $[\Delta_W] = [-1, 1]$ mm for each checkerboard corner. This calibration target was positioned in a distance of approximately 1 m to 2 m in front of the multi-sensor system.

8.1.2.2 Derivation of sensor error bounds

In Section 4.2 we introduced bounded error models for the IMU and the camera. These models require error bounds for every different type of error we identified. Naturally, the error bounds differ from sensor to sensor. Thus, in the following we derive or determine these bounds for the sensors introduced in the previous section that are then employed for all following experiments.

IMU

According to Section 4.2.3, we need to determine the uncertainties of the scale factor $[\Delta_s]$, the constant bias $[\boldsymbol{\Delta}_b]$, the walking bias $[\boldsymbol{\Delta}_{wb}]$, and the measurement noise $[\boldsymbol{\Delta}_n]$.

Fortunately, the manufacturer of our IMU provides a comprehensive data sheet in which all uncertainties are listed. First, the scale factor stability is given as $[\Delta_s] = [-0.05, 0.05]\,\%$. Second, the manufacturer specifies the constant bias as $[\boldsymbol{\Delta}_b] = ([-0.05, 0.05]\,^\circ/\mathrm{s})^\intercal_{\times 3}$. Third, the walking bias is given as $5\,^\circ/\mathrm{h}$, and thus we set $[\boldsymbol{\Delta}_{wb}] = ([-0.0014, 0.0014]\,^\circ/\mathrm{s}^2)^\intercal_{\times 3}$. Fourth and last, the data sheet states a measurement noise density of $0.002\,^\circ/\mathrm{s}/\sqrt{\mathrm{Hz}}$. Because the IMU is operated at a bandwidth of 100 Hz, the noise density is $0.02\,^\circ/\mathrm{s}$. Since this is the standard deviation of the noise, which is assumed to be Gaussian, we choose bounds that enclose the true noise with a probability of 99.7 %, and thus we set $[\boldsymbol{\Delta}_{wb}] = ([-0.03, 0.03]\,^\circ/\mathrm{s})^\intercal_{\times 3}$.

Camera

According to Section 4.2.2, we need to determine intervals enclosing the feature detection error caused by quantization $[\Delta_{px_q}]$, by image blur $[\Delta_{px_b}]$ and by measurement noise $[\Delta_{px_n}]$.

Usually, $[\Delta_{px_q}] = [-0.5, 0.5]$ px since the actual scene is discretized into pixels. However, the detection of checkerboard corners can be more accurate since multiple observations (i.e. intersection of lines) are used to find a single corner. In contrast, image blur and measurement noise cannot be quantified just as easily since many different factors - some of which are unknown - have an impact. Consequently, we can only choose error bounds $[\Delta_{px}]$ that summarize all of the previously mentioned types of error. Since we only need to specify the maximum error occurring during the detection of the checkerboard corners, we determined the maximum reprojection error occurring during the intrinsic camera calibration and employ it as the interval error bounds for the camera features: $[\Delta_{px}] = [-0.3, 0.3]$ px.

8.1.2.3 Results

Overall, we collected four different data sets of different characteristics. For the first trial, we added a motor to our setup such that we could conduct an experiment with a rotation around only one axis. For this experiment, we set the velocity of the motor to 1 rad/s. The following three data sets were recorded while rotating the setup by hand and ensuring that the 3D target remained in the field of view of the camera. At first, however, we placed our setup on the floor in front of the 3D target to define the reference coordinate system, picked it up after roughly 1 s and rotated it for approximately 10 s. We aimed to increase the rotational velocity from trial to trial, such that the second trial contains low velocity rotations and the fourth trial contains high velocity rotations.

Time offset only

Similar to our simulation studies, we solve the one-dimensional problem at first, which means that we use a fixed interval vector for the rotation and only employ the time offset contractor $\mathcal{C}_{\text{offset}}$ to compute an interval enclosing the time offset between camera and IMU. The three-dimensional MRP vector is estimated by hand and different measurement uncertainties are added.

Trial	$[\tau_1]$ (ms)	$[\tau_2]$ (ms)	τ_{xcorr} (ms)
1	$[41.0, 68.4]$	$[37.1, 73.2]$	50.9
2	$[1.0, 94.7]$	$[-23.4, 126.0]$	48.8
3	$[25.4, 76.2]$	$[2.0, 102.5]$	48.2
4	$[29.3, 60.5]$	$[5.9, 77.1]$	45.8

Table 8.4: Results of $\mathcal{C}_{\text{offset}}$ to only compute the time offset $[\tau]$ for different real experiments.

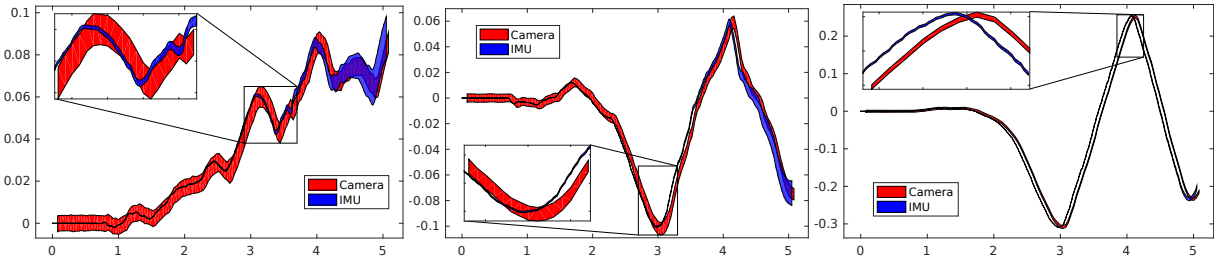
The results can be seen in Table 8.4. We choose a rather small rotation uncertainty of $w([\theta_C^I]) = 6°$ for the column containing $[\tau_1]$ and a large rotation uncertainty of $w([\theta_C^I]) = 18°$ for the column containing $[\tau_2]$. As can be seen, the interval widths, and thus the accuracies are similar to our experiments using simulated data.

Furthermore, as expected after the simulation studies, the accuracy with which we are able to enclose the time offset again depends on the velocity with which we rotated our sensor

setup. However, it makes no difference whether the setup is rotated around all three or just one axis. Furthermore, all computed intervals overlap, which shows that the results are consistent and the calibration is repeatable. Since the results should be guaranteed, it is inevitable that all computed intervals share a common intersection. If not, we would have discovered an inconsistency. The common intersection for our experiments is $[\tau] = [41.0, 60.5]$ ms. Thus, the final uncertainty for our one-dimensional studies is $w([\tau]) = 19.5$ ms. Naturally, this accuracy can be improved by estimating more accurate rotation parameters.

To provide some reference data, we compute the orientation of the camera and the IMU over time using traditional methods and apply cross-correlation to temporally align the data series as described by Mair et al. [127]. The results, which can be seen in the fourth column of Table 8.4, are always enclosed in the corresponding interval for the timestamp offset. Consequently, the results of our interval-based method to compute the time offset between camera and IMU are in accordance with the results of an established approach.

Finally, Fig. 8.3 provides a visualization of the time offset between camera and IMU by showing the respective orientation tubes of camera and IMU for the fourth trial after aligning them spatially using the estimated bounded rotation.



(a) $[\rho_x](\cdot)$. (b) $[\rho_y](\cdot)$. (c) $[\rho_z](\cdot)$.

Figure 8.3: Spatially aligned orientation tubes (rotation over time) for the fourth trial that are non-overlapping due to a timestamp offset.

Full spatiotemporal calibration

Next, we solve the four-dimensional problem, meaning that we use our time offset contractor $\mathcal{C}_{\text{offset}}$ in combination with SIVIA to enclose both the three-dimensional MRP interval vector for the extrinsic rotation and the time offset between camera and IMU. We choose the same initial rotation uncertainty of $w([\theta]) = 124^\circ$ as for our simulation studies. Moreover, we set an accuracy of $\epsilon = 0.001$ for the SIVIA algorithm.

As Table 8.5 shows, we are not able to estimate the rotation to a reasonable accuracy for the first trial in which we employed the motor to rotate around one axis only. Nevertheless, we manage to contract the three-dimensional MRP interval vectors for trials two to four. While the results are comparable to the results we achieve using simulated data, they are too inaccurate to be employed for fusing data from camera and IMU as will be explained later. We believe the reason for this inaccuracy is two-fold. On the one hand, the rapidly increasing uncertainty of the orientation tubes of the IMU prevents us from gathering sufficient constraints. After just a few seconds, the uncertainty is too large due to drift. On the other hand, we are limited in our movement of the multi-sensor system because the calibration target must be kept in

Trial	$[\tau]$ (ms)	$[\rho_x]$	$[\rho_y]$	$[\rho_z]$	$w([\theta])$ (°)
1	$[43.0, 65.4]$	$[-0.42, -0.21]$	$[-0.50, -0.10]$	$[-0.50, -0.10]$	100.72
2	$[27.3, 68.4]$	$[-0.35, -0.31]$	$[-0.35, -0.32]$	$[-0.36, -0.30]$	12.77
3	$[35.2, 57.6]$	$[-0.35, -0.31]$	$[-0.35, -0.32]$	$[-0.36, -0.30]$	12.24
4	$[37.1, 51.8]$	$[-0.36, -0.31]$	$[-0.36, -0.32]$	$[-0.36, -0.30]$	12.93

Table 8.5: Results of the full spatiotemporal calibration for different real experiments that vary by the rotation axis and the rotational velocity. The single components of the MRP are denoted as follows: $[\boldsymbol{\rho}_C^I] = (\,[\rho_x]\;\;[\rho_y]\;\;[\rho_z]\,)^\intercal$.

the field of view of the camera. Consequently, we cannot perform large rotations which could introduce additional constraints for the contraction of the extrinsic rotation.

In contrast, the computation of the time offset between camera and IMU is reasonably accurate and can be used for sensor fusion. Again, the intervals for the offset share a common intersection $[\tau] = [43.0, 51.8]$ ms ms, which is also consistent with the result of our one-dimensional studies. Thus, the final accuracy we achieve is $w([\tau]) = 8.8$ ms.

In contrast to the time offset contractor $\mathcal{C}_{\text{offset}}$, which requires less than 0.1 s computation time, the computation time for the four-dimensional problem ranges from 10 min for the fourth trial to 1 h for the second trial. The computation time for the first trial amounts to 10 h since the rotation parameters do not converge. However, since our approach is supposed to be employed for an offline calibration prior to the actual experiment, computation time should play a subordinate role compared to the calibration accuracy.

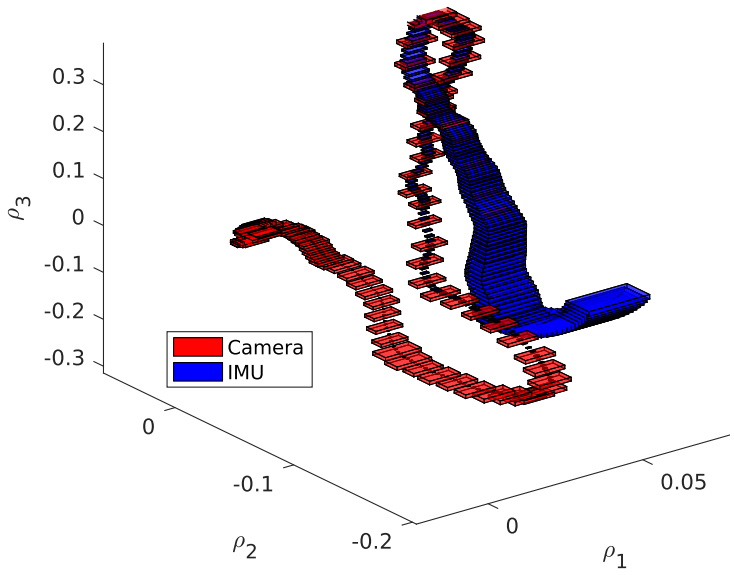


Figure 8.4: Aligned camera and IMU orientation intervals after calibration.

Fig. 8.4 shows the camera and IMU orientation intervals which were aligned using the computed extrinsic rotation. Note the increasing uncertainty of the IMU orientation due to the integration of the angular velocities.

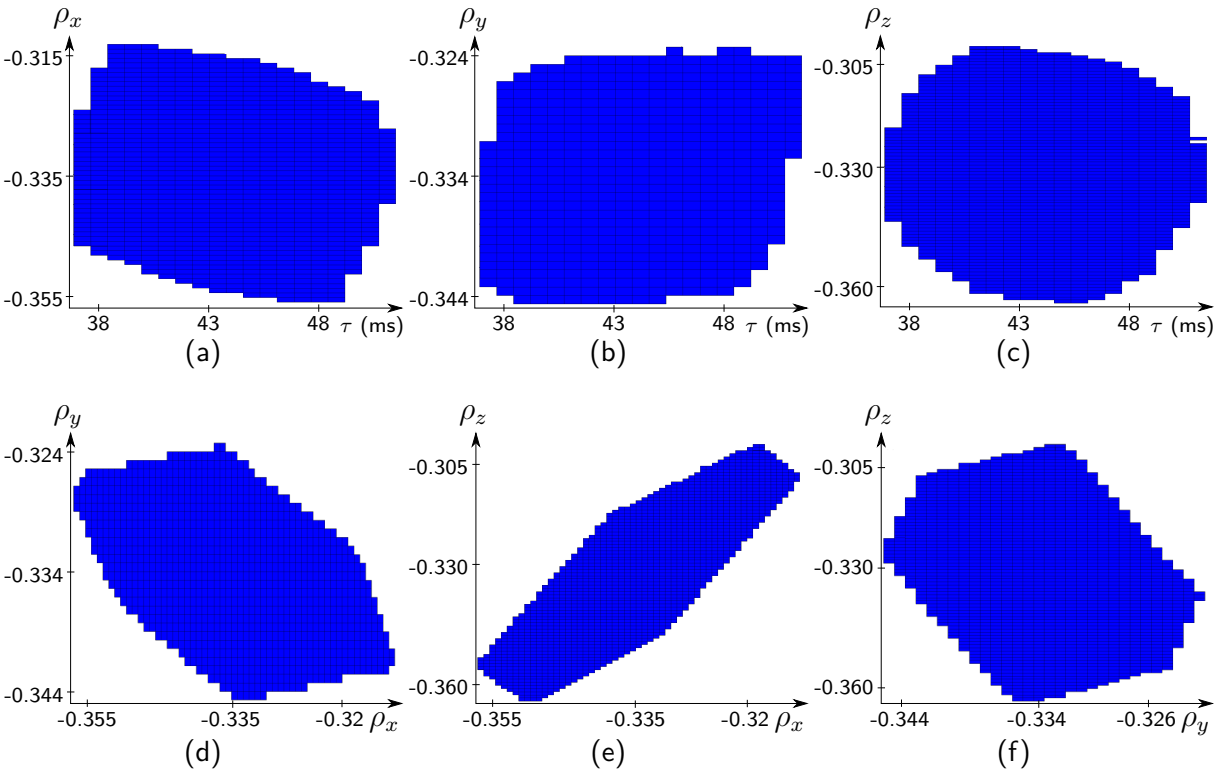


Figure 8.5: Subpavings corresponding to the fourth trial. The single components of the MRP are denoted as follows: $[\boldsymbol{\rho}_C^I] = (\,[\rho_x]\;\;[\rho_y]\;\;[\rho_z]\,)^\intercal$. Figures have been drawn using VIBes [124].

Finally, we depict the subpaving (cf. Section 3.8.2) corresponding to the fourth trial in Fig. 8.5. Since it is impossible to display a four-dimensional subpaving, we decided to show the dependencies between each two pairs. The graphs should be interpreted as follows. Consider Fig. 8.5e. $[\rho_x]$ is plotted on the x-axis pointing to the right and $[\rho_z]$ is plotted on the y-axis pointing to the top. The possible combinations of $[\rho_x]$ and $[\rho_z]$ that belong to the solution set are colored blue. This means that $[\rho_x]$ can only be large if $[\rho_z]$ is large as well (i.e. in the top right corner). However, there is no solution if $[\rho_x]$ is small and $[\rho_z]$ is large (i.e. in the top left corner).

It becomes evident that the timestamp offset $[\tau]$ does not depend on the rotation parameters. This can be observed by the fact that the upper subpavings are close to rectangular. In contrast, there exist dependencies between the rotation parameters themselves. Especially the subpaving in Fig. 8.5e is not rectangular, thus indicating that we are not able to compute $[\rho_x]$ independent of $[\rho_z]$. This can be attributed to an insufficient rotation by hand. As previously explained, we did not manage to rotate around all three axis at the same time while keeping the 3D target in the field of view of the camera.

8.2 Extrinsic calibration between camera and LiDAR

This section details the experimental evaluation of the approach for the extrinsic calibration between camera and LiDAR that we presented in Section 6.2. The main evaluation criterion is the desired guarantee of being able to reliably enclose the true solution. Since it is difficult to find the true solution using real data, we first evaluate the approach using simulated data before showing that it is also capable of dealing with real data. In addition, we compare the accuracies of different results for different sensor error bounds and show the advantages of our method over an established approach.

If not stated otherwise, we set the following accuracies for the different instances of the SIVIA algorithm we employ. First, the accuracy of the SIVIA algorithm fitting a plane to laser scan boxes (cf. Section 6.2.2) is set to $\epsilon_{\text{plane}} = 0.001$. Second, the accuracy of the SIVIA algorithm employed to solve the PnP problem (cf. Section 5.2) is set to $\epsilon_{\text{pnp}} = 0.001°$. Third, the accuracy of the SIVIA algorithm for the final computation of the extrinsic transformation parameters (cf. Section 6.2.3) is set to $\epsilon_{\text{calibration}} = 0.1°$.

8.2.1 Simulated data

Finding the transformation between camera and laser scanner for a real multi-sensor system without relying on sensor data is infeasible since the sensors are generally black-box systems that do not allow to find the coordinate system origin. Thus, any computed transformation is compromised by data errors and we have no means to asses the true transformation between the sensors. Hence, to evaluate our approach against ground truth data, we have to resort to simulated data. Furthermore, this enables us to simulate bounded errors and the number of outliers in a guaranteed way, allowing us to evaluate these influences on the calibration result. As a prerequisite for the experiments carried out, the experimental environment used is first explained.

8.2.1.1 Simulated experimental environment

To simulate the experiment, we again employed the multi-robot simulator *Gazebo* [125]. Since the assumption made in Section 6.2.2 states that there should be no interfering objects around the checkerboard target, we constructed a simulation environment that is solely composed of the checkerboard and the two sensors. Recreating this environment in the real world can be done by performing the experiment in a laboratory. At first, we added the checkerboard that is constructed according to the real world object we use for later experiments. The total size of the board is 100×76 cm, while the imprinted checkerboard is composed of 10 x 7 squares with a side length of 8 cm each. We placed the checkerboard in a distance of roughly 2.5 m from the multi-sensor system.

We used Gazebo's predefined camera controller to simulate the camera. Unfortunately, it only allows to simulate Gaussian noise for each frame and does not provide an option to define the error distribution manually. Nevertheless, it enabled us to evaluate our approach for different mean errors and different standard deviations for which we defined the interval error bounds according to Section 3.1. We simulated an error following a Gaussian distribution

with a mean value of 0 and a standard deviation of 0.01. This noise value was added to each pixel's color channels which lie in the range from 0 to 1. We derived the bounded error for the camera as explained in Section 4.2.2 and set $[\Delta_{px}] = [-0.3, 0.3]$ px. Furthermore, the controller allowed us to manually set the parameters of the pinhole camera model (cf. Section 2.1.2). We set similar parameters as we encounter for the real camera - i.e. we set an image resolution of 1920×1080 px and $f_x = f_y = 2200$ px, $c_x = 960$ px, $c_y = 540$ px.

To simulate the laser scanner, we defined an own controller that is based on Gazebo's ray sensor. Our controller replicates the laser scanner we use for our experiments and allows to set unknown but bounded errors for the spherical coordinates of each scan point. In total, we simulated 300.000 points per second in a 360° horizontal and a 30° vertical field of view. The laser scanner is set to rotate with a frequency of 5 Hz, resulting in a horizontal angular resolution of 0.1° and a vertical angular resolution of 2° (16 individual lasers/channels). To simulate the unknown but bounded errors, we applied a random number generator following a uniform distribution. For the first experiments, we simulated no outliers and the following bounded errors: $[\Delta_r] = [-3, 3]$ cm and $[\Delta_\theta] = [\Delta_\varphi] = [-0.03, 0.03]°$ (cf. Section 4.2.1).

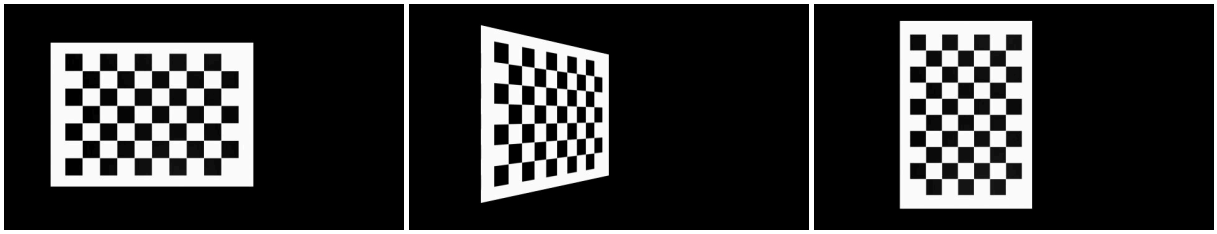
The true rotation matrix $\mathbf{R}_L^C$ between camera and laser scanner is expressed using Euler angles (cf. Section 2.3.1). For our first experiments, we set $\phi_L^C = 90°$, $\theta_L^C = 0°$ and $\psi_L^C = 0°$. Furthermore, the translation is set to $\mathbf{T}_L^C = ({}_xT_L^C \;\; {}_yT_L^C \;\; {}_zT_L^C)^\intercal = (-27\,\text{cm} \;\; 15\,\text{cm} \;\; -12\,\text{cm})^\intercal$. We chose these transformation parameters to approximately replicate our real multi-sensor system. Later, we also show results for different transformation parameters to show that our approach encloses the true parameters regardless of their characteristics. Finally, we set $\epsilon_{\text{calibration}} = 0.1°$ for the first simulated experiments.

8.2.1.2 Results from individual checkerboard poses

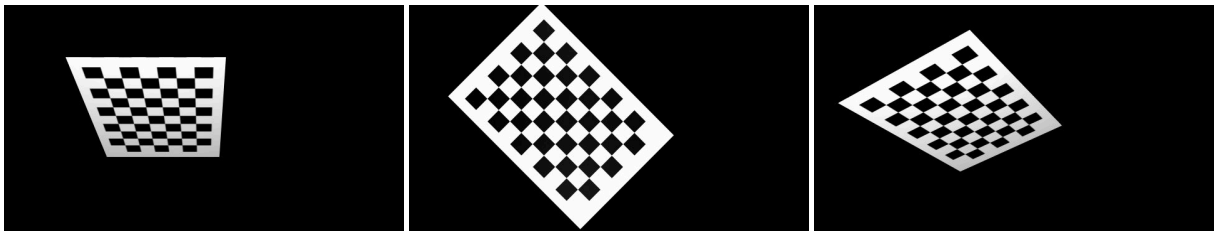
At first, we show results for the transformation parameters that are computed from one checkerboard pose only. As explained in Section 6.2, one sensor data pair (i.e. one camera image and one laser scan) suffices to perform the extrinsic calibration. However, the accuracy is expected to be different from one transformation parameter to another since one checkerboard pose does not enable us to constrain all six parameters equivalently. We select six different checkerboard poses, which are depicted in Fig. 8.6, to show the influences on different extrinsic calibration parameters. Furthermore, the initial domains for the rotation are set to $[\theta_L^C] = [-90, 90]°$ and $[\phi_L^C] = [\psi_L^C] = [-180, 180]°$. Besides, we set $[{}_wT_L^C] = {}_wT_L^C + [-50, 50]$ cm for $w \in \{x, y, z\}$. Thus, we assume no initial information about the extrinsic transformation other than a rough idea of the setup of the multi-sensor system. This is necessary since some parameter domains cannot be contracted at all.

Table 8.6 shows the results and Table 8.7 depicts the corresponding interval widths. As can be seen, all intervals enclose the true result, showing the expected behavior of our approach. Nevertheless, we can observe some interesting characteristics. While all six checkerboard poses allow to compute a reasonably accurate domain for the three Euler angles, some translation domains cannot be contracted for some poses. However, also the rotation parameters differ from one pose to another. This can be explained by two different phenomena.

The first phenomenon is related to the geometry of the individual checkerboard poses. For example, the fifth pose does not allow an accurate computation of the rotation around the



(a) Pose 1. (b) Pose 2. (c) Pose 3.



(d) Pose 4. (e) Pose 5. (f) Pose 6.

Figure 8.6: Simulated camera images of six different checkerboard poses which we employed individually to perform the extrinsic calibration between camera and LiDAR.

Pose	$[\phi_L^C]$ (°)	$[\theta_L^C]$ (°)	$[\psi_L^C]$ (°)	$[{}_xT_L^C]$ (cm)	$[{}_yT_L^C]$ (cm)	$[{}_zT_L^C]$ (cm)
1	$[89.3, 90.7]$	$[-0.6, 0.5]$	$[-0.6, 0.7]$	$[-29.7, -24.4]$	$[-35.0, 65.0]$	$[-13.6, -10.5]$
2	$[88.9, 91.2]$	$[-0.4, 0.5]$	$[-0.8, 0.7]$	$[-29.9, -24.4]$	$[-35.0, 65.0]$	$[-13.9, -9.9]$
3	$[89.3, 90.8]$	$[-0.9, 0.9]$	$[-0.4, 0.5]$	$[-31.5, -22.4]$	$[-35.0, 65.0]$	$[-13.8, -10.1]$
4	$[89.6, 90.3]$	$[-0.6, 0.6]$	$[-0.7, 0.7]$	$[-30.5, -24.1]$	$[-17.4, 47.0]$	$[-62.0, 38.0]$
5	$[88.9, 91.1]$	$[-1.1, 1.1]$	$[-0.1, 0.3]$	$[-32.5, -21.2]$	$[9.1, 20.6]$	$[-13.6, -10.6]$
6	$[89.2, 90.7]$	$[-1.1, 0.9]$	$[-1.1, 1.2]$	$[-32.0, -21.2]$	$[11.2, 18.6]$	$[-14.2, -9.5]$

Table 8.6: Results using **single** simulated checkerboard poses to constrain the transformation parameters. The pose identifiers correspond to the visualizations in Fig. 8.6.

x- and y-axis, but prevails at contracting the interval for the rotation around the z-axis. This is due to the fact that the boundary line direction vectors lie in the x-y-plane of the camera coordinate system, and thus a rotation around the z-axis has a larger effect than a rotation around the x- or y-axis. While this is also true for the first pose, the fifth pose enables us to identify all four boundary lines, and thus also the corner points, resulting in more information and a tighter enclosure of the rotation around the z-axis. However, in contrast, the first pose is better suited to constrain the rotation around the x-axis since its detectable boundary lines also reside in the y-z-plane.

Nevertheless, this does not explain why the first pose results in a tighter enclosure for the rotation around the y-axis. This rotation is mainly constrained by the plane normal vector, which is geometrically the same for both poses. However, as the checkerboard is rotated with respect to the camera coordinate system for the fifth pose, the computation of the PnP problem suffers from the wrapping effect (cf. Section 3.8.1), and thus the pose estimate of the camera with respect to the checkerboard is less accurate. However, this pose estimate is

Pose	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
1	1.4	1.1	1.3	5.3	100.0	3.1
2	2.3	0.9	1.5	5.5	100.0	4.0
3	1.5	1.8	0.9	9.1	100.0	3.7
4	0.7	1.2	1.4	6.4	64.4	100.0
5	2.2	2.2	0.4	11.3	11.5	3.0
6	1.5	2.0	2.3	10.8	7.4	4.7

Table 8.7: Interval widths for the results from Table 8.6.

required to compute the plane normal vector in the camera coordinate system, and thus the estimation of the plane normal vector becomes less accurate as well. In contrast, the pose estimate of the camera with respect to the checkerboard is more accurate for the first pose, resulting in a more accurate computation of the plane normal vector, and thus in stronger constraints for the rotation around the y-axis.

However, the geometry alone does not explain why the rotation around the x-axis is most constrained by the fourth pose. But, this can be explained by the second phenomenon that applies for every interval analysis approach and is explained in Section 3.11. In short, the better the error bounds represent the actual error we encounter for a sensor, the more accurate the results become. Since the fourth pose leads to larger image errors, and thus allows the PnP problem to be solved more accurately, the resulting features required for the extrinsic calibration are more accurate as well. Consequently, as the geometry of the pose is best suited to constrain the rotation around the x-axis, the corresponding interval can be contracted more tightly. In contrast, the image errors are small for the first pose due to the best possible viewing angle, and thus the PnP problem is solved less accurately. However, as we have to choose conservative error bounds that enclose all possible values, this behavior is expected.

After discussing the results for the rotation parameters, we want to focus on the translation parameters in the following. As already mentioned, some poses do not provide sufficient constraints to contract the corresponding intervals. For example, the first pose does not enable our approach to compute the translation along the y-axis (downwards). When looking at the camera image in Fig. 8.6a and the CSP (6.37) it becomes obvious why this is the case. Since two of the boundaries of the checkerboard (top and bottom) are parallel to the scan lines of the laser scanner, we are not able find border points on these boundaries. Consequently, it becomes impossible to compute corner points since we can only determine the right and left boundary lines of the checkerboard in the coordinate system of the laser scanner. Thus, only the third and fourth constraint of the CSP (6.37) constrain the translation $\mathbf{T}_L^C$. When looking at these constraints in detail, it becomes evident that the third constraint forces border points to lie on the boundary lines and the fourth constraint forces points to lie on the checkerboard plane. Since the two detectable boundary lines are parallel to the y-axis in the camera coordinate system, the third constraint only constrains the translation in x- and z-direction, but allows arbitrary movements in y-direction. Similarly, the checkerboard plane is

parallel to the x-y-plane in the camera coordinate system, and thus it follows that the fourth constraint only constrains the translation in z-direction. In summary, the first checkerboard pose does not provide constraints on the translation in y-direction.

In contrast, the fourth pose does provide - albeit weak - constraints on the translation along the y-axis. This is due to the fact that we rotate the checkerboard such that the plane becomes almost parallel to the x-z-plane in the camera coordinate system. However, this comes at the cost of not being able to contract the translation along the z-axis. The explanation for this behavior is the same as before. However, the constraints are weaker since - as conceivable from Fig. 8.6d - less scan points reside on the checkerboard (i.e. less information is available) and the checkerboard plane is not exactly parallel to the x-z-plane. Rotating the checkerboard even further is not possible, as the laser scanner would not see it anymore.

Furthermore, the fifth and sixth pose allow to contract the domains for all six transformation parameters. However, some of the computed intervals (e.g. $[\theta_L^C]$) are less accurate due to the fact that less border points can be computed for each checkerboard boundary, and thus less information is available to find the direction vectors of the boundary line. Besides, some of the increasing uncertainty can be attributed to the wrapping effect (cf. Section 3.8.1) since the boundaries of the checkerboard are not parallel to the coordinate system of the laser scanner. However, when trying to compute the extrinsic calibration parameters from one pose, it is advisable to employ a checkerboard pose similar to the fifth or sixth pose for the best possible accuracy.

8.2.1.3 Results from multiple checkerboard poses

After showing the results for contracting the extrinsic calibration parameter domains based on single checkerboard poses, we depict the results for the same algorithm that combines the constraints of all six checkerboard poses. However, as we are confident that the combination of these poses allows us to contract all six parameter domains, we employ different initial domains. The initial domains for the rotation are set to $[\theta_L^C] = [-90, 90]^\circ$ and $[\phi_L^C] = [\psi_L^C] = [-180, 180]^\circ$. Besides, we set $[{}_wT_L^C] = {}_wT_L^C + [-\infty, \infty]$cm for $w \in \{x, y, z\}$. Thus, we assume no initial information for the extrinsic calibration parameters.

Table 8.8 outlines the results and the corresponding interval widths. As expected, the computed transformation parameter domains resemble the best possible results of employing the different poses individually. For example, when intersecting the domains for $[\theta_L^C]$ from the first and second pose, we can deduct that $[\theta_L^C] = [-0.4, 0.5]^\circ$, which is consistent with the result we compute when combining all six poses. However, some domains - such as the translation parameters - can be determined even more accurately. This can be explained by the fact that the combined constraints supplement each other. For example, the contractor built for the sixth checkerboard pose manages to contract the translation in y-direction, which in turn leads the contractor built for the first pose to contract the translation in x- and z-direction even more accurately. Thus, this experiment shows a textbook example of how different contractors can be combined to produce an even more powerful contractor.

Having shown reasonably accurate results when employing six checkerboard poses, we want to examine whether significant improvements in accuracy can be achieved if even more checkerboard poses are used. In total, we employ 27 different checkerboard poses, which are

x	$[\phi_L^C]$ (°)	$[\theta_L^C]$ (°)	$[\psi_L^C]$ (°)	$[{}_xT_L^C]$ (cm)	$[{}_yT_L^C]$ (cm)	$[{}_zT_L^C]$ (cm)
$[x]$	$[89.6, 90.3]$	$[-0.4, 0.5]$	$[-0.1, 0.3]$	$[-29.6, -25.0]$	$[12.7, 17.0]$	$[-13.1, -11.0]$
$w([x])$	0.7	0.9	0.4	4.6	4.3	2.1

Table 8.8: Results combining all six (cf. Fig. 8.6) simulated checkerboard poses to constrain the transformation parameters.

displayed in Fig. 8.7. The poses are selected such that many different orientations of the checkerboard are covered, while making sure that both camera and laser scan data are suitable to identify the required features. As explained in Section 6.2, moving the checkerboard is not necessary since this does not produce additional constraints.

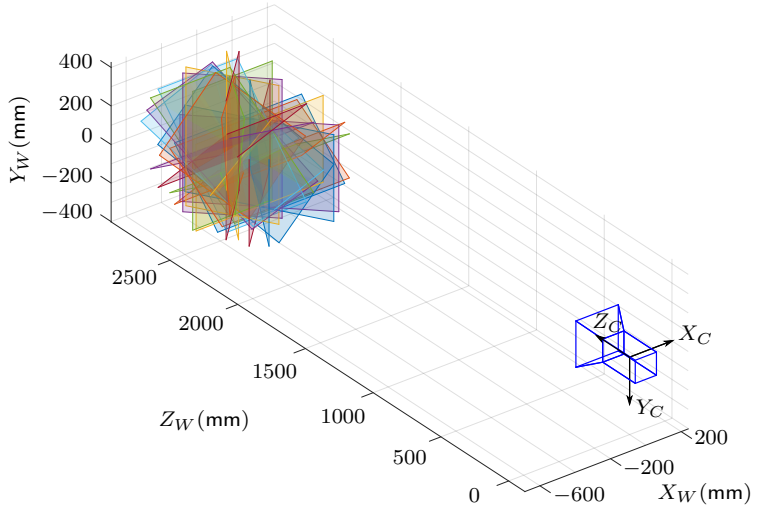


Figure 8.7: Visualization of 27 different checkerboard poses which are simultaneously employed to contract the extrinsic transformation parameter domains.

Table 8.9 shows the results and the corresponding interval widths. It is evident that the additional checkerboard poses do not enable us to contract transformation parameter domains much further. The largest improvements can be seen for the translation in x- and y- direction, for which we reduce the uncertainty by 0.8 cm (17.4 %) and 0.7 cm (16.3 %), respectively. However, this comes at the cost of an additional effort for the calibration since more data has to be collected and the computation takes more time.

x	$[\phi_L^C]$ (°)	$[\theta_L^C]$ (°)	$[\psi_L^C]$ (°)	$[{}_xT_L^C]$ (cm)	$[{}_yT_L^C]$ (cm)	$[{}_zT_L^C]$ (cm)
$[x]$	$[89.6, 90.3]$	$[-0.4, 0.3]$	$[-0.1, 0.3]$	$[-28.8, -25.0]$	$[13.1, 16.7]$	$[-13.1, -11.0]$
$w([x])$	0.7	0.7	0.4	3.8	3.6	2.1

Table 8.9: Results combining all 27 (cf. Fig. 8.7) simulated checkerboard poses to constrain the transformation parameters.

An interesting observation that can be made is that both the rotation around and the translation in direction of the z-axis can be computed more accurately than the other rotation

and translation parameters. This can be explained by analyzing the constraints of the CSP (6.37) and the geometry of possible checkerboard poses. As mentioned previously, the fourth constraint of the CSP forces scan points to reside on the checkerboard plane. Thus, it induces strong constraints on the translation in the direction of the plane normal vector. Consequently, the best possible constraints can be established for the coordinate system axis that is parallel to the plane normal vector. However, it is not possible to position the checkerboard such that the x- or y-axis of the camera coordinate system coincides with the plane normal vector of the checkerboard since this would not allow the detection of the features required for the extrinsic calibration (i.e. the checkerboard in Fig. 8.6b and Fig. 8.6d would have to be rotated even further in the corresponding direction, resulting in an image of the checkerboard in side view). In contrast, placing the checkerboard such as for Fig. 8.6a makes the plane normal vector parallel to the z-axis of the camera coordinate system. The reason why the fourth constraint is stronger than the other constraints involving the translation is that scan points residing on the checkerboard plane can be computed more accurately and in a larger number than border points (third constraint) or corner points (fifth constraint).

Similarly, the strongest constraints on the rotation parameters are imposed by the second constraint of the CSP (6.37) since we compute multiple line direction vectors that are generally more accurate than the plane normal vector (first constraint). Geometrically, a direction vector is particularly suited to constrain the rotation around the plane normal vector of the plane it resides in (e.g. a direction vector in the x-y-plane imposes strong constraints on the rotation around the z-axis). Thus, according to the same argument as in the previous paragraph, the rotation around the z-axis can be computed the most accurate since we can only find line direction vectors in the x-y-plane (cf. Fig. 8.6e).

8.2.1.4 Results for different bisection widths

After showing the general applicability of our approach for the extrinsic calibration of a simulated multi-sensor system that resembles the true setup we will use for our real experiments, we want to evaluate the influence of different parameters on the calibration result.

We start by investigating the importance of the parameter $\epsilon_{\text{calibration}}$, which specifies the bisection width, and thus the accuracy of the final SIVIA algorithm that computes the domains for the transformation parameters. In order to do that, we varied $\epsilon_{\text{calibration}}$ and observed the resulting computation time as well as the accuracy of the transformation parameter. To achieve the best possible accuracy, we employed all 27 checkerboard poses. Table 8.10 shows the results. Note that the reported computation times concern only the final SIVIA algorithm for which the required camera and laser scanner features have been extracted previously. The computation time required to extract these features varies for each checkerboard pose, totaling 240 s for all 27 checkerboard poses combined.

As can be seen, $\epsilon_{\text{calibration}}$ does not significantly influence the accuracy of all six transformation parameters. While we observe a slight improvement for the width of the translation parameter domains when decreasing $\epsilon_{\text{calibration}}$ from 1.0° to 0.05°, the computation time drastically increases from 1 s to 1121 s since a significantly larger number of bisections have to be performed. This result shows that the forward-backward contractor alone is already strong enough to contract the transformation parameter domains and does not rely on additional bisections. Nevertheless,

$\epsilon_{\text{calibration}}$ (°)	Computation time (s)	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
1.0	1	0.70	0.78	0.47	3.92	3.71	2.20
0.1	155	0.70	0.78	0.47	3.85	3.68	2.04
0.05	1121	0.70	0.78	0.47	3.83	3.67	2.01

Table 8.10: Influence of the extrinsic calibration parameter $\epsilon_{\text{calibration}}$, which specifies the bisection width, and thus the accuracy of the SIVIA algorithm, on both the computation time and the accuracy of the parameters. The reported computation times refer only to the final SIVIA algorithm.

it is still possible to slightly improve the result by setting $\epsilon_{\text{calibration}} = 0.05°$. Since the extrinsic calibration between camera and LiDAR is usually performed once before using the resulting domains for many subsequent experiments, it is worthwhile to invest additional computation time - although this may only slightly improve the accuracy.

8.2.1.5 Results for different errors

In the following, we depict calibration results for different simulated laser scanner and camera errors. For the first experiment, we simulated different errors for the three spherical coordinates of each scan point (cf. Section 4.2.1). As explained in the beginning of this section, we add an error to each of these parameters that is randomly sampled inside the chosen error bounds. Instead of using all 27 checkerboard poses, we fall back on the six poses depicted in Fig. 8.6 since these poses were sufficient to achieve a reasonable accuracy. Furthermore, we set $\epsilon_{\text{calibration}} = 1.0$ to reduce computation time.

Simulated errors			Results					
$r([\Delta_r])$ (cm)	$r([\Delta_\theta])$ (°)	$r([\Delta_\varphi])$ (°)	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
3.0	0.03	0.03	0.70	0.92	0.47	4.72	4.51	2.23
6.0	0.03	0.03	0.75	1.10	0.71	5.52	4.74	2.29
3.0	0.06	0.03	0.81	1.02	0.67	5.22	5.19	2.25
3.0	0.03	0.06	0.71	0.98	0.81	5.20	4.88	2.26
6.0	0.06	0.06	0.92	1.25	0.95	6.19	5.84	2.42

Table 8.11: Influence of different simulated laser scanner errors on the accuracies of the extrinsic transformation parameters.

Table 8.11 depicts the results. In principle this rule applies: increasing the error for any of the three spherical parameters results in an increasing uncertainty of the extrinsic transformation parameters. However, the uncertainty does not increase significantly, indicating that our approach can cope with different errors when the appropriate error bounds are known.

In addition, we also simulated different errors for the checkerboard corner feature detections. Originally, we simulated an error for the camera by adding Gaussian noise to each pixel's color channels. However, increasing this noise does not automatically result in a corresponding increase of the pixel error with which the checkerboard corners are identified. Therefore, we added an additional artificial error to each checkerboard corner detection instead. Accordingly, the interval error bounds $[\Delta_{px}]$ are adapted to enclose this additional error. Since the original corner detections already introduce an error of $[-0.3, 0.3]$ px in the worst case, adding an additional artificial error that is randomly sampled, for example, in the interval $[-0.1, 0.1]$ px results in an error of $[-0.4, 0.4]$ px in the worst case. Since the interval error bounds have to reflect this worst case, we set $[\Delta_{px}] = [-0.4, 0.4]$ px for this example.

$r([\Delta_{px}])$ (px)	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
0.3	0.70	0.92	0.47	4.72	4.51	2.23
0.4	0.77	0.94	0.51	4.83	4.89	2.69
0.5	0.82	0.96	0.54	5.01	5.23	2.99
0.6	0.82	1.06	0.55	5.46	5.30	3.20

Table 8.12: Influence of different simulated camera errors on the accuracy of the parameters.

Table 8.12 outlines the pixel error for the checkerboard corner detections and the resulting width of the extrinsic calibration parameters. Again, increasing the error results in wider intervals reflecting the increasing uncertainty. However, similar to the previous experiment, the uncertainty does not increase significantly. Thus, we can state that our approach can handle different errors for both camera and laser scan data without drastically increasing the uncertainty of the resulting transformation parameters. Of course, this holds only on the condition that the error bounds are known and not overestimated, since otherwise the uncertainty can increase sharply as explained in Section 3.11.

8.2.1.6 Results under consideration of outliers

Next, we introduced artificial outliers in the laser scan data to assess the robustness of our approach with respect to outliers. We did not add outliers to the camera images since the employed checkerboard detector can reliably detect the checkerboard corner features and does not introduce large errors which could be classified as outliers. To add outliers to the laser scan data, we first selected a maximum outlier percentage $\delta_{\text{max outliers}}$ which resulted in a maximum number of outliers out of the N_p scan points on the checkerboard: $n_{\text{max outliers}} = \delta_{\text{max outliers}} \cdot N_p$. Consequently, we randomly picked $n_{\text{outliers}} \in \{0, \ldots, n_{\text{max outliers}}\}$ to represent the fact that we do not know the actual number of outliers for our real experiments, but only an upper bound. Afterwards, we randomly selected n_{outliers} scan points and modified their distance measurement such that it is no longer consistent with the error bounds we previously chose.

Subsequently, we employ a q-relaxed intersection for the laser scan feature extraction as explained in Section 6.2.2. Due to the design of our experiment, $n_{\text{max outliers}}$ outliers can occur at most, and thus we set $q = n_{\text{max outliers}}$ for the q-relaxed intersection (cf. Section 3.9).

Again, we only use the six poses depicted in Fig. 8.6 and set $\epsilon_{\text{calibration}} = 1.0$ to reduce computation time. Since the accuracy drastically varies with the actual numbers of outliers n_{outliers}, we repeat the experiment 100 times and report the average accuracy over all 100 runs. Table 8.13 shows the results for different outlier percentages $\delta_{\text{max outliers}}$. Since all results enclose the true transformation parameters, we only present the resulting interval widths.

$\delta_{\text{max outliers}}$ (%)	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
0.0	0.70	0.92	0.47	4.72	4.51	2.23
0.1	0.70	0.97	0.48	4.91	4.52	2.24
0.5	0.86	1.08	0.49	5.42	5.21	2.33
1.0	0.95	1.16	0.50	5.78	5.66	2.38
2.0	1.16	1.26	0.53	6.22	6.53	2.53
5.0	1.54	1.45	0.61	7.07	8.11	2.92

Table 8.13: Influence of laser scan outliers on the accuracy of the parameters. The depicted interval widths are the average interval widths over 100 consecutive runs. For each run, **at most** $\delta_{\text{max outliers}} \cdot N_p$ outliers are randomly generated in the laser scan data.

The results show that the uncertainties of the parameters increase with the maximum outlier percentage $\delta_{\text{max outliers}}$. However, we claim that this phenomenon is not related to the actual outliers, but to the fact that we generated $\delta_{\text{max outliers}} \cdot N_p$ **at most**, and often the actual number of outliers is smaller than this. Thus, we overestimate the number of outliers. This decreases the accuracy with which we can extract the features from the laser scan data, and thus increases the uncertainty of the entire extrinsic calibration.

To verify this claim, we show results for which n_{outliers} is not randomly picked, but automatically set to $n_{\text{outliers}} = n_{\text{max outliers}}$. Thus, the number of actual outliers we encounter complies with the parameter we feed into the q-relaxed intersection. Table 8.14 shows the outcome. As can be seen, the results support our claim since the interval widths of the parameters change only slightly (0.01° at most) with an increasing outlier percentage. This change can be attributed to the fact that the outliers influence the bisection process during the laser scanner feature extraction. As there are some outliers, there are less valid constraints to extract the plane parameters from the set of scan points. Thus, to achieve the desired accuracy during the SIVIA algorithm more bisections are required, which can then even increase the accuracy of the plane parameters.

However, these results are only useful to explain where the increasing uncertainty in Table 8.13 stems from. When employing real data, it is impossible to know how many outliers occur, and thus we can only specify a maximum number of outliers to expect. Nevertheless, this also highlights the importance of not overestimating the actual number of outliers, as doing so can lead to an increasing uncertainty of the calibration parameters.

$\delta_{\text{max outliers}}$ (%)	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
0.0	0.70	0.92	0.47	4.72	4.51	2.23
0.1	0.70	0.92	0.47	4.72	4.51	2.23
0.5	0.70	0.93	0.47	4.72	4.51	2.23
1.0	0.70	0.92	0.47	4.71	4.51	2.23
2.0	0.70	0.92	0.47	4.71	4.51	2.23
5.0	0.70	0.92	0.47	4.71	4.50	2.23

Table 8.14: Influence of laser scan outliers on the accuracy of the parameters. For this experiment, **exactly** $\delta_{\text{max outliers}} \cdot N_p$ outliers are randomly generated in the laser scan data.

8.2.1.7 Results for different true transformations

For our final simulated experiment, we created different true transformations between laser scanner and camera to ensure that our approach works for arbitrary transformations and to evaluate the influence of the transformation on the accuracy of the parameter domains. Note, that the transformations are not truly arbitrary, as it is mandatory to find checkerboard poses for which both the camera and the laser scanner can extract the required features with a reasonable accuracy. We employed two different transformations for this experiment. To create these, we kept the pose of the laser scanner constant and moved the camera around. This has the advantage that the same 27 checkerboard poses from Fig. 8.7 can be employed for the computation since they were originally chosen to replicate all checkerboard poses for which we can extract features from laser scan data. However, as the camera is moved, we cannot extract features from all 27 camera images, and thus some poses were discarded. Furthermore, we set $\epsilon_{\text{calibration}} = 1.0$ to reduce computation time. Again, we set initial domains for the rotation to $[\theta_L^C] = [-90, 90]°$ and $[\phi_L^C] = [\psi_L^C] = [-180, 180]°$ and for the translation to $[{}_wT_L^C] = {}_wT_L^C + [-\infty, \infty]$cm for $w \in \{x, y, z\}$.

x	ϕ_L^C (°)	θ_L^C (°)	ψ_L^C (°)	${}_xT_L^C$ (cm)	${}_yT_L^C$ (cm)	${}_zT_L^C$ (cm)
x^*	100.345	14.767	2.664	-53.5	65.1	5.0
$[x]$	$[99.9, 100.8]$	$[14.2, 15.3]$	$[2.0, 3.2]$	$[-56.6, -50.4]$	$[61.9, 67.9]$	$[2.3, 7.7]$
$w([x])$	0.9	1.1	1.2	6.2	6.0	5.4
x^*	71.887	-33.644	10.272	151.6	-31.8	55.2
$[x]$	$[71.2, 72.7]$	$[-34.4, -32.9]$	$[9.2, 11.3]$	$[147.1, 155.5]$	$[-36.4, -27.0]$	$[51.8, 58.6]$
$w([x])$	1.5	1.5	2.1	8.4	9.4	6.8

Table 8.15: Evaluation using two different true extrinsic transformations between camera and laser scanner.

The results are depicted in Table 8.15. They show that our approach reliably encloses the true transformation parameters. However, the domain widths, and thus the accuracies of the parameters vary. This can be observed for both the upper transformation and the lower transformation. Nevertheless, a plausible explanation for this behavior can be given by considering the wrapping effect (cf. Section 3.8.1). For the previous experiments each coordinate system axis of the camera coordinate system was parallel to a coordinate system axis of the laser scanner coordinate system. Thus, the wrapping effect did not occur for the transformation parameters. In contrast, the coordinate system axes of camera and laser scanner are rotated with respect to each other, inducing an additional uncertainty due to the wrapping effect. This applies especially for the second transformation since this transformation incorporates a large rotation, but is also true for the first transformation. In particular, the uncertainty for the rotation around the z-axis and the translation along the z-axis increases. Previously, a checkerboard pose for which the normal vector is parallel to axes of both the camera and laser scanner coordinate system allowed us to contract these transformation parameters tightly. However, due to the rotation between camera and laser scanner coordinate system, this is no longer possible.

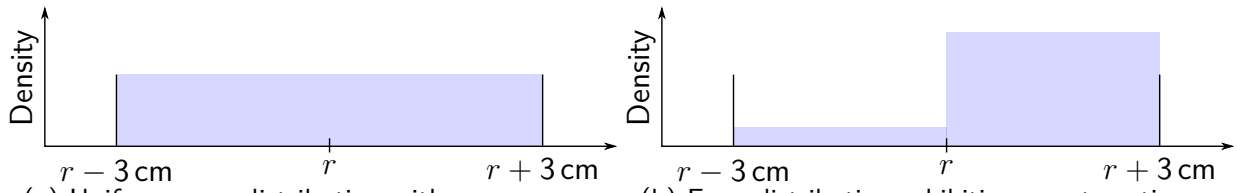
Nevertheless, it can be seen that the uncertainty of the translation parameters does not increase proportionally with their absolute value. For example, the percentage uncertainty of the translation along the x-axis lies in the interval $[13.2, 15.2]\,\%$ for the transformation employed in the previous experiments, in the interval $[11.0, 12.3]\,\%$ for the first transformation and in the interval $[5.4, 5.7]\,\%$ for the second transformation. Moreover, the translation uncertainty increases along the y-axis for the second transformation, although its absolute value decreases with respect to the first transformation. Thus, it is evident that the translation uncertainty does not scale proportionally with the absolute value, but is affected by the rotation and the induced wrapping effect.

8.2.1.8 Comparison to traditional approach

To show the differences and highlight the advantages of our approach to state-of-the-art methods, we compare it to the approach of Zhou et al. [49]. However, a direct comparison is infeasible since the objectives of both approaches are different. On the one hand, Zhou et al. aim to compute point-valued parameters for the extrinsic transformation between camera and laser scanner while ignoring the underlying uncertainties. On the other hand, our approach is designed to consistently propagate the sensor errors to the extrinsic transformation parameters, but cannot provide point-valued results. Nevertheless, we show the advantages of our approach that are revealed in the presence of systematic errors (i.e. biases) and during the selection of unfavorable checkerboard poses.

Influence of systematic errors

Since Zhou et al. refrain from modeling sensor errors, their approach only works in the presence of zero-mean errors. However, systematic errors violating this prerequisite can occur for both camera and laser scanner as explained in Section 4.1. In contrast, our approach can cope with systematic errors since they are modeled implicitly when relying on interval analysis. The following experiment, for which the results are depicted in Table 8.16, illustrates this fact.



(a) Uniform error distribution with zero-mean error.

(b) Error distribution exhibiting a systematic error (bias) of 1 cm.

Figure 8.8: Two different error distributions we employ to show the influence of systematic errors in the laser distance measurements.

We employ the same simulated 27 checkerboard poses as in Section 8.2.1.3. Originally, the error follows a uniform distribution in the interval $[-0.03, 0.03]$ m (cf. Fig. 8.8a), and thus does not violate the zero-mean assumption of the state-of-the-art approach. The corresponding results for both approaches are depicted in the second and third row of Table 8.16. As can be seen, the state-of-the-art approach performs reasonably well and is able to accurately estimate the extrinsic transformation parameters. Furthermore, our own approach exhibits the same characteristics as explained in Section 8.2.1.3.

Subsequently, we added a bias to the error of the distance measurements of the laser scanner such that the mean distance error is 1 cm, but the error is still bounded in the interval $[-0.03, 0.03]$ m. Fig. 8.8b shows the corresponding error distribution. The fourth and fifth row of Table 8.16 show the corresponding extrinsic calibration parameters. As can be seen, the approach of Zhou et al. [49] is significantly influenced by the systematic error. As expected, the translation along the z-axis is off by approximately 1 cm, as this is the forward facing axis, which is thus most affected by a distance error of the laser scanner. In contrast, our approach is not disturbed by systematic errors and the results are similar to the results for the unbiased data.

	ϕ_L^C (°)	θ_L^C (°)	ψ_L^C (°)	${}_xT_L^C$ (cm)	${}_yT_L^C$ (cm)	${}_zT_L^C$ (cm)
True	90.0	0.0	0.0	-27.0	15.0	$\mathbf{-12.0}$
[49], no bias	90.01	-0.01	-0.04	-27.00	14.96	-11.91
Our, no bias	$[89.6, 90.3]$	$[-0.4, 0.3]$	$[-0.1, 0.3]$	$[-28.8, -25.0]$	$[13.1, 16.7]$	$[-13.1, -11.0]$
[49], bias	89.98	0.01	0.04	-27.09	14.88	$\mathbf{-12.97}$
Our, bias	$[89.7, 90.3]$	$[-0.4, 0.5]$	$[-0.4, 0.3]$	$[-29.5, -25.0]$	$[13.0, 16.8]$	$\mathbf{[-13.1, -10.9]}$

Table 8.16: Results showing the influence of biased distance measurements on both our and the stochastic approach.

Selection of checkerboard poses

Another problem arising during the extrinsic calibration of camera and laser scanner is the selection of appropriate and sufficient checkerboard poses. Thus, Zhou et al. [49] introduce their method with the ulterior motive of reducing the minimal number of poses required for the calibration to one. Their approach, however, lacks the capability to determine if the chosen

checkerboard poses are sufficient, as sensor errors are neglected and thus not propagated to the final calibration result. Consequently, the user cannot determine whether additional checkerboard poses are required to improve the calibration accuracy. The following experiment illustrates this limitation.

As for the evaluation of our approach, we employ the six checkerboard poses introduced in Section 8.2.1.2 individually to perform the extrinsic calibration using the approach of Zhou et al. [49]. Table 8.17 shows the results. Furthermore, Table 8.18 recalls the resulting interval widths of our approach for the same data. As can be seen, the state-of-the-art method cannot compute the correct extrinsic calibration parameters from every individual pose. For example, the translation along the y-axis is incorrect for the poses one to three. While this is the expected behavior, as these poses do not provide sufficient constraints to compute this parameter, the results can be misleading because the user cannot recognize these inaccuracies from the results alone. Although our approach obviously cannot compute these parameters either (i.e. the intervals do not get contracted), the results reflect the uncertainty and allow to recognize the need for additional checkerboard poses.

Pose	ϕ_L^C (°)	θ_L^C (°)	ψ_L^C (°)	${}_xT_L^C$ (cm)	${}_yT_L^C$ (cm)	${}_zT_L^C$ (cm)
True	90.0	0.0	0.0	−27.0	15.0	−12.0
1	89.43	−0.21	0.04	−26.21	0.00	−12.11
2	89.89	0.01	0.06	−26.98	−349.18	−11.94
3	90.27	0.42	−0.01	−28.95	0.00	−11.89
4	89.83	0.05	0.04	−27.33	15.66	−9.30
5	89.64	−1.37	0.07	−20.43	13.12	−11.58
6	89.94	−0.22	0.00	−25.99	14.47	−12.15

Table 8.17: Results from applying the approach of Zhou et al. [49] to **single** simulated checkerboard poses. The pose identifiers correspond to the visualizations in Fig. 8.6.

Pose	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
1	1.4	1.1	1.3	5.3	100.0	3.1
2	2.3	0.9	1.5	5.5	100.0	4.0
3	1.5	1.8	0.9	9.1	100.0	3.7
4	0.7	1.2	1.4	6.4	64.4	100.0
5	2.2	2.2	0.4	11.3	11.5	3.0
6	1.5	2.0	2.3	10.8	7.4	4.7

Table 8.18: Interval widths from applying our approach to **single** simulated checkerboard poses. Full results are depicted in Table 8.6.

(a) Exemplary calibration image which depicts the calibration environment as well as the employed checkerboard.

(b) Multi-sensor system, consisting of a camera and a laser scanner, that is employed to evaluate the extrinsic calibration.

Figure 8.9: Overview of the equipment used for the extrinsic calibration.

Furthermore, the fifth and sixth checkerboard pose allow our approach to constrain all six calibration parameters. However, the interval widths indicate that the results are still inaccurate and data from additional poses should be added. Similarly, the approach of Zhou et al. manages to compute an approximation for all six parameters. However, the translation along the x-axis, which also our approach identified as the most inaccurate translation parameter, still exhibits an error of more than 1 cm. Thus, additional checkerboard poses are required to improve the accuracy.

Since capturing data of a variety of checkerboard poses is a cumbersome task, it is beneficial to require as few poses as possible. Nevertheless, the accuracy of the extrinsic calibration parameters should not suffer from the selection of too few checkerboard poses. Under this perspective, our approach allows to immediately assess the calibration accuracy to determine if additional checkerboard poses are necessary and to learn which poses constrain which calibration parameters. As depicted, this is in contrast to the presented state-of-the-art approach.

8.2.2 Real data

After confirming the correctness of our extrinsic calibration approach for simulated data, we show the applicability to real data. However, as finding ground truth information is infeasible, we can only evaluate the calibration accuracy (i.e. the interval widths) and compare the results to the results computed using the approach of Zhou et al. [49]. As a prerequisite for the experiments carried out, the experimental environment is depicted.

8.2.2.1 Real experimental environment

Fig. 8.9a shows an exemplary image captured by the camera during the extrinsic calibration. As can be seen, the checkerboard is mounted on a tripod that is positioned in free space such that we can define a box containing nothing but the checkerboard. As explained earlier, this allows us to easily determine the set of scan points residing on the checkerboard. It remains to

filter out the scan points on the tripod stands. However, this is straightforward as there are only a few that can be easily identified with RANSAC.

Next, we introduce the sensors employed for the experiments. Fig. 8.9b shows the setup consisting of laser scanner and camera that are rigidly mounted on aluminum profiles. The camera is a Point Grey Grasshopper3 USB3 vision camera that we operated at a resolution of 1920×1200 px. The focal length, which we determined by performing a intrinsic camera calibration using the same images we acquired for the extrinsic calibration, amounts to approximately 2200 px. The intrinsic calibration is performed using the algorithm proposed by Zhang [17] and the corresponding implementation in OpenCV. Subsequently, we used the computed distortion parameters to remove distortion from all images.

The laser scanner is a Velodyne PUCK (VLP-16) with a total of 16 channels. It has a range of 100 m and generates approximately 300.000 points per second, which lie in a horizontal field of view of 360° and a vertical field of view of 30°. Consequently, the vertical angular resolution is 2°. We set the rotation rate of the laser scanner to 5 Hz, resulting in a horizontal angular resolution of 0.1°.

For our experiments, we positioned the checkerboard, which has a total size of 100×76 cm, in a distance of approximately 2 m to 3 m in front of the multi-sensor system.

8.2.2.2 Derivation of sensor error bounds

In Section 4.2 we introduced bounded error models for the camera and the laser scanner. These models require error bounds for every different type of error we identified. Naturally, the error bounds differ from sensor to sensor. Thus, in the following we derive or determine these bounds for the sensors introduced in the previous section that are then employed for all following experiments.

Laser scanner

According to Section 4.2.1, we need to determine the uncertainties of the distance measurement $[\Delta_r]$, the vertical angle $[\Delta_\theta]$, the horizontal angle $[\Delta_\varphi]$ and the dimensions of the initial footprint $[\Delta_{b_\theta}]$ and $[\Delta_{b_\varphi}]$.

Regarding the distance uncertainty $[\Delta_r]$, the manufacturer specifies an accuracy of ± 3 cm. Consequently, we choose $[\Delta_r] = [-3, 3]$ cm. Next, the manufacturer specifies the horizontal divergence of a laser beam as 3 mrad and the vertical beam divergence as 1.5 mrad. Thus, we set $[\Delta_\varphi] = [-1.5, 1.5]$ mrad and $[\Delta_\theta] = [-0.75, 0.75]$ mrad. Last, the dimensions of the initial footprint of the laser beam are given as 9.5 mm tall by 12.7 mm wide. Consequently, we choose $[\Delta_{b_\theta}] = [-4.75, 4.75]$ mm and $[\Delta_{b_\varphi}] = [-6.35, 6.35]$ mm.

Furthermore, we assume a maximum outlier percentage of $\delta_{\text{max outliers}} = 0.5\,\%$ for the computation of the plane parameters, which we determined empirically.

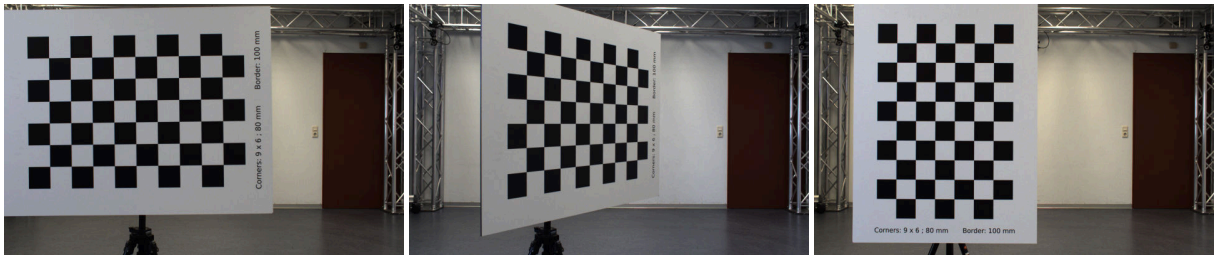
Camera

The derivation of the sensor error bounds for the camera that we employ to evaluate the extrinsic calibration approach can be found in Section 8.1.2.2, since we employ the same sensor

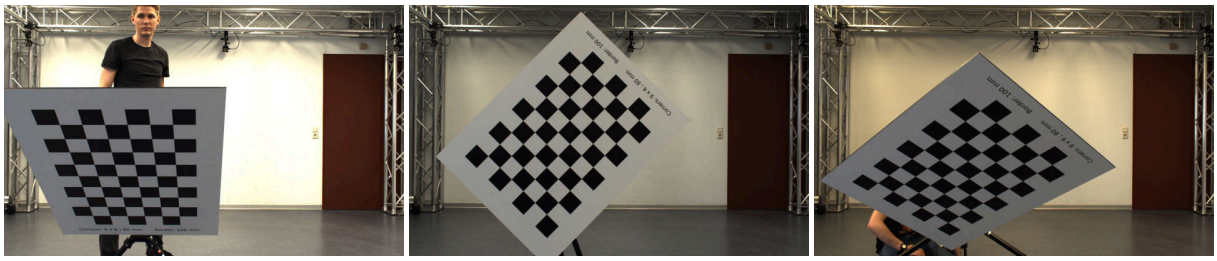
to measure the same data as in this section. For this experiment, the maximum reprojection error during camera calibration is 0.5 px, and thus we set $[\Delta_{px}] = [-0.5, 0.5]$ px. As before, we cannot precisely access the printing accuracy of the checkerboard, and thus we conservatively assume $[\Delta_W] = [-1, 1]$ mm for each checkerboard corner.

8.2.2.3 Results from individual checkerboard poses

In the same way as for the simulated data, we first show results from individual checkerboard poses. Similarly, we expect different checkerboard poses to constrain different transformation parameters with varying degrees of accuracy. We choose six different checkerboard poses that resemble the poses we simulated in Section 8.2.1.2. Fig. 8.10 shows the corresponding images taken by the camera. Since we expect each individual pose to provide sufficient constraints to contract the rotation parameters - although with different accuracies - we set the initial domains to $[\theta_L^C] = [-90, 90]°$ and $[\phi_L^C] = [\psi_L^C] = [-180, 180]°$, and thus assume no initial information about them. In contrast, not every checkerboard pose is suited to contract all three translation parameters. Assuming no information about them would result in infinite uncertainties which would prevent us from contracting any parameter domains. Thus, we set $[\mathbf{T}_L^C] = (\,[-77, 23]\,\text{cm}\;\; [-35, 65]\,\text{cm}\;\; [-62, 38]\,\text{cm}\,)$, which constitutes a basic assumption about the multi-sensor system that can be verified during a visual inspection.



(a) Pose 1. (b) Pose 2. (c) Pose 3.



(d) Pose 4. (e) Pose 5. (f) Pose 6.

Figure 8.10: Camera images of six different checkerboard poses which we employed individually to perform the extrinsic calibration between camera and LiDAR.

Table 8.19 shows the results and Table 8.20 depicts the corresponding interval widths. We created similar checkerboard poses as in our simulation studies, and as can be seen, the results are therefore similar to our simulation studies. For example, the first three poses again do not provide constraints to contract the translation along the y-axis. Moreover, the accuracies for the three Euler angles again differ in the same way as in our simulation studies. For example, the second pose results in the tightest contraction of the rotation around y-axis. The same

Pose	$[\phi_L^C]$ (°)	$[\theta_L^C]$ (°)	$[\psi_L^C]$ (°)	$[{}_xT_L^C]$ (cm)	$[{}_yT_L^C]$ (cm)	$[{}_zT_L^C]$ (cm)
1	$[88.5, 91.4]$	$[-1.6, 0.6]$	$[-0.3, 1.6]$	$[-30.2, -20.4]$	$[-35.0, 65.0]$	$[-13.1, -9.6]$
2	$[87.7, 92.0]$	$[-0.7, 0.8]$	$[-0.3, 2.4]$	$[-30.0, -23.5]$	$[-35.0, 65.0]$	$[-14.3, -7.7]$
3	$[88.3, 90.7]$	$[-1.5, 1.1]$	$[0.0, 1.6]$	$[-31.8, -20.7]$	$[-35.0, 65.0]$	$[-13.6, -9.4]$
4	$[89.8, 90.8]$	$[-1.2, 1.2]$	$[-0.9, 2.0]$	$[-32.6, -21.5]$	$[-26.7, 61.8]$	$[-62.0, 38.0]$
5	$[87.2, 92.1]$	$[-2.4, 2.7]$	$[0.0, 1.0]$	$[-39.5, -16.1]$	$[4.2, 25.8]$	$[-14.4, -8.5]$
6	$[89.1, 91.2]$	$[-1.9, 0.9]$	$[-0.9, 2.3]$	$[-31.1, -19.2]$	$[13.5, 22.4]$	$[-14.2, -8.5]$

Table 8.19: Results using **single** checkerboard poses to constrain the transformation parameters. The pose identifiers correspond to the visualizations in Fig. 8.10.

Pose	$w([\phi_L^C])$ (°)	$w([\theta_L^C])$ (°)	$w([\psi_L^C])$ (°)	$w([{}_xT_L^C])$ (cm)	$w([{}_yT_L^C])$ (cm)	$w([{}_zT_L^C])$ (cm)
1	2.9	2.2	1.9	9.8	100.0	3.5
2	4.3	1.5	2.7	6.5	100.0	6.6
3	2.4	2.6	1.6	11.1	100.0	4.2
4	1.0	2.4	2.9	11.1	88.5	100.0
5	4.9	5.1	1.0	23.4	21.4	5.9
6	2.1	2.8	3.2	11.9	8.9	5.7

Table 8.20: Interval widths for the results from Table 8.19.

applies to the fourth pose and the rotation around the x-axis, and the fifth pose and the rotation around the z-axis.

However, it can also be seen that the overall accuracy is lower than in our simulation studies. For example, using simulated data the fifth pose allows us to compute the rotation around the y-axis with an accuracy of $w([\theta_L^C]) = 2.2°$, while we are only able to reach an accuracy of $w([\theta_L^C]) = 5.1°$ using real data. The same is true for every pose and every transformation parameter. Thus, in the following we provide an explanation for this finding. First, we observed that the accuracies of the checkerboard features (i.e. plane parameters, line parameters and corner points) extracted from camera images are similar for real and simulated data. In contrast, we are able to compute these checkerboard features more accurately using simulated instead of real point clouds. Therefore, the lower accuracy of the extrinsic calibration must be related to the checkerboard feature extraction from the point clouds.

As explained for our simulation studies, we simulate the error of the laser scanner to follow a uniform distribution. Consequently, the error bounds we chose in our simulation studies are optimal since they are guaranteed to enclose the true error, but do not overestimate it. In contrast, the error bounds for our experiments using real data are chosen to comply with the information provided by the manufacturer. However, the actual error may not exhaust the error bounds, and is therefore overestimated. As a result, the checkerboard features are extracted

less accurately, which in turn leads to a less tight contraction of the extrinsic transformation parameters.

8.2.2.4 Results from multiple checkerboard poses

After presenting the results computed from single checkerboard poses, we combine the contractors constructed from multiple checkerboard poses to determine more accurate transformation parameter domains. Since combining those contractors allows us to contract the domains of all six transformation parameters, we can assume no initial information about them and set $[{}_wT_L^C] = [-\infty, \infty]$cm for $w \in \{x, y, z\}$. As before, we also set $[\theta_L^C] = [-90, 90]^\circ$ and $[\phi_L^C] = [\psi_L^C] = [-180, 180]^\circ$. At first, we combine the constraints from the six checkerboard poses we presented in the previous section.

x	$[\phi_L^C]$ (°)	$[\theta_L^C]$ (°)	$[\psi_L^C]$ (°)	$[{}_xT_L^C]$ (cm)	$[{}_yT_L^C]$ (cm)	$[{}_zT_L^C]$ (cm)
$[x]$	$[89.8, 90.7]$	$[-0.7, 0.6]$	$[0.2, 1.0]$	$[-29.0, -24.2]$	$[15.6, 20.0]$	$[-12.6, -10.2]$
$w([x])$	0.9	1.3	0.8	4.8	4.4	2.4

Table 8.21: Results combining all six (cf. Fig. 8.10) checkerboard poses to constrain the transformation parameters.

Table 8.21 presents the results and the corresponding interval widths. As in our simulation studies, it is evident that we are able to compute the translation along the z-axis more accurately. Overall, the interval widths are similar, although not quite as tight as for our simulated six checkerboard poses. However, we were able to reduce the comparatively large uncertainty of our previous evaluation, in which we employed individual checkerboard poses to constrain the transformation parameters.

In the following, we further extend the number of constraints by employing a total of 26 distinct checkerboard poses for which we rotated the checkerboard in front of the multi-sensor system. Some poses are selected such that we can just barely extract the desired features from both images and point clouds while maximizing the rotation around one or several axes. This allows stronger constraints to be found. However, moving (i.e. performing a translation) the checkerboard is not required as detailed in Section 6.2. Fig. 8.11 depicts those 26 checkerboard poses relative to the coordinate system of the camera.

x	$[\phi_L^C]$ (°)	$[\theta_L^C]$ (°)	$[\psi_L^C]$ (°)	$[{}_xT_L^C]$ (cm)	$[{}_yT_L^C]$ (cm)	$[{}_zT_L^C]$ (cm)
$[x]$	$[89.8, 90.5]$	$[-0.6, 0.4]$	$[0.4, 1.0]$	$[-28.3, -25.2]$	$[15.7, 18.8]$	$[-12.5, -10.3]$
$w([x])$	0.7	1.0	0.6	3.1	3.1	2.2
x' [49]	90.6	-0.6	0.9	-24.6	15.7	-13.2

Table 8.22: Results combining all 26 (cf. Fig. 8.11) checkerboard poses to constrain the transformation parameters.

Table 8.22 shows the results and the corresponding interval widths. By using additional checkerboard poses in addition to the six previously presented, we can further reduce the

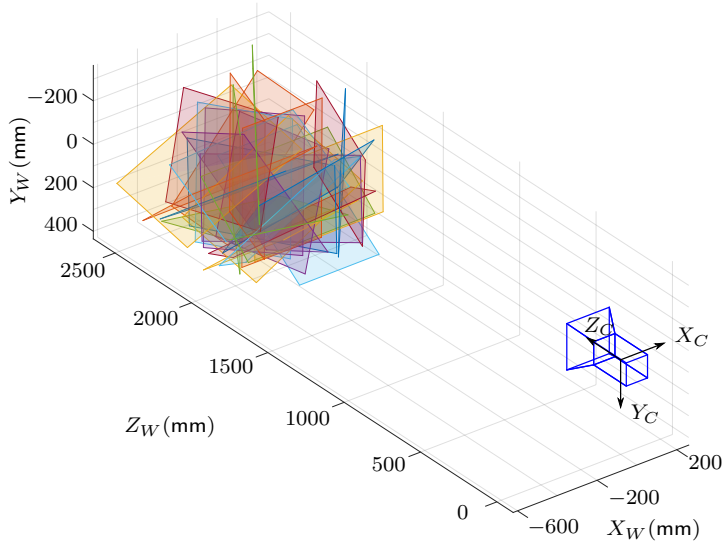


Figure 8.11: Visualization of 26 different checkerboard poses which are simultaneously employed to contract the extrinsic transformation parameter domains.

uncertainty of the extrinsic transformation parameters. Again, it can be seen that the resulting interval widths are similar to our simulation studies, in which we used 27 checkerboard poses.

Since the results using real data are similar to using simulated data, we can conclude that our approach is suitable to find the extrinsic transformation between camera and laser scanner, and can be applied to real data. Moreover, the intervals should be guaranteed to enclose the true parameters, since we were able to reliably enclose the true parameters using simulated data and the accuracies are comparable using real data.

In addition to the results of our approach, Table 8.22 contains the results of the established stochastic approach introduced by Zhou et al. [49]. It can be seen that some of the computed parameters are not enclosed by our corresponding intervals. However, since our intervals are guaranteed to contain the true solution, the results of the stochastic approach cannot be accurate. A possible reason could be systematic errors that are not compatible with the assumption of zero-mean errors in stochastic approaches, as highlighted in Section 8.2.1.8. Consequently, our approach can be used not only to perform the extrinsic calibration, but also to determine whether the parameters computed by a stochastic approach could be correct. Moreover, our approach enables the final accuracy of the calibration to be assessed, and is therefore able to determine whether additional checkerboard poses are required to improve the results. In order to further compare our interval-based approach with the stochastic approach in the future, various metrics such as the reprojection error could be used. Moreover, a different setup consisting of two cameras and the laser scanner could be employed to use the stereo camera calibration as ground truth information.

8.3 Bounded-error visual-LiDAR odometry

This section details the experimental evaluation of the approach for interval-based visual-LiDAR odometry that we presented in Chapter 7. The main evaluation criterion is the desired guarantee of being able to reliably enclose the true solution. Consequently, we require a dataset that contains accurate ground truth information. In the following, we present such a dataset that was gathered within the Research Training Group i.c.sens[2]. Afterwards, we introduce the sensor error bounds used for the evaluation and detail how they were derived. We then present the results, which include different evaluation criteria such as accuracy or number of outliers. Finally, we examine results using different error bounds or different parameters to show the effect of these.

8.3.1 Dataset

Since our approach for visual-LiDAR odometry is dedicated to autonomous driving in urban areas, we require data from sensors attached to a vehicle that is driving through a city. For this purpose, a so-called *mapathon* was organized by our research training group i.c.sens [128]. Here, we equipped a van with a multi-sensor platform that can be seen in Fig. 8.12. Among other sensors, this platform houses a LORD MicroStrain 3DM-GQ4-45 GNSS aided IMU, a Velodyne HDL-64E laser scanner and a FLIR Grasshopper3 GS3-U3-23S6C-C color camera. The IMU was operated at a measurement frequency of 500 Hz.

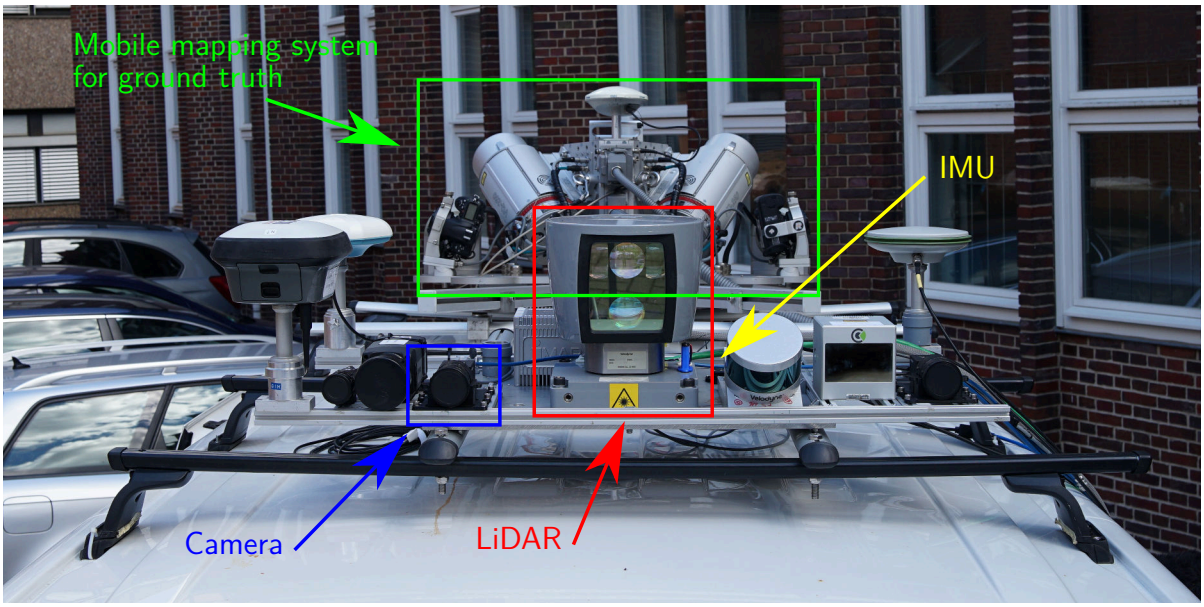


Figure 8.12: Image of our multi-sensor platform on which the sensors relevant for this work were marked. Red: Velodyne HDL-64E laser scanner, blue: FLIR Grasshopper3 color camera, yellow (hidden behind other sensors): LORD MicroStrain IMU, green: Riegl VMX-250 Mobile Mapping System. Image credit: Sören Vogel.

The laser scanner was set to rotate with a frequency of 10 Hz resulting in a horizontal angular resolution of 0.1728°. It consists of 64 laser beams firing simultaneously that are

[2]The Research Training Group i.c.sens [RTG 2159] is funded by the German Research Foundation (DFG).

arranged on top of each other resulting in a vertical angular resolution of 0.5°. Thus, the Velodyne HDL-64E laser scanner delivers a vertical field of view from 2° to −29.5°.

The camera was equipped with a Fujinon CF12.5HA-1 lens that has a focal length of 12.5 mm. Unfortunately, due to the different vertical fields of view of the camera and the laser scanner, no distance information can be assigned by the laser scanner to the image features in the upper third of the image. This can be seen in Fig. 7.2. We define the coordinate system of the camera, which is the reference coordinate system for our odometry computations, as follows. The z-axis is pointing in the viewing direction, the x-axis is pointing to the right and the y-axis is pointing down.

Moreover, the image size of the camera is 1920×1200 px and it is set to capture images whenever the laser scanner faces forward, thus resulting in a frame rate of 10 frames per second. For this synchronization of camera and laser scanner, we employed the idea depicted in [129]. Here, the authors propose to use a reed contact attached to the rotating laser scanner to generate an electrical pulse that triggers the camera whenever the laser scanner is facing forward. Consequently, the camera and laser scanner are directly synchronized in time. In addition, the electrical pulse was also routed into a Raspberry Pi that is synchronized to Global Positioning System (GPS) time using an EVK-M8T GNSS receiver and the GPS daemon (gpsd). Thus, we were able to assign an accurate GPS timestamp to every image and laser scan point. Since our IMU encompasses a GNSS receiver, the timestamps of the measured angular velocities are also given in the GPS time reference, and thus all our sensors are synchronized via GPS time.

To measure accurate ground truth information, the vehicle was additionally equipped with a Riegl VMX-250 Mobile Mapping System that consists of two laser scanners, a camera system and a localization unit. The Mobile Mapping System can be seen in the back of Fig. 8.12. For this work, we are particularly interested in the localization unit that consists of a highly accurate GNSS/IMU system coupled with an external distance measurement instrument (Applanix POS LV 510).

To determine the extrinsic calibration between the Mobile Mapping System and our own platform, a laser tracker (Leica Absolute Tracker AT960) was used to accurately measure reference points on the sensor housings. Moreover, to enable a more accurate extrinsic calibration (compared to our own spatiotemporal calibration approaches) between camera, laser scanner and IMU, the laser tracker was employed to measure known reference planes and control points in a laboratory, which were then identified in the images and point clouds.

For the evaluation of this work, we choose a dataset that was acquired in the "Nordstadt" of Hanover, Germany. The test environment and the route are depicted in Fig. 8.13. As can be seen, there are many tall buildings along the route that prevent GNSS information but offer many visual features. However, there are also passages with some vegetation and thus less rich image features, but better GNSS reception. In total, we recorded data for 262 s and covered a distance of 1328 m during this timespan.

8.3.2 Derivation of sensor error bounds

In Section 4.2 we introduced bounded error models for the camera, the laser scanner and the IMU. These models require error bounds for every different type of error we identified.

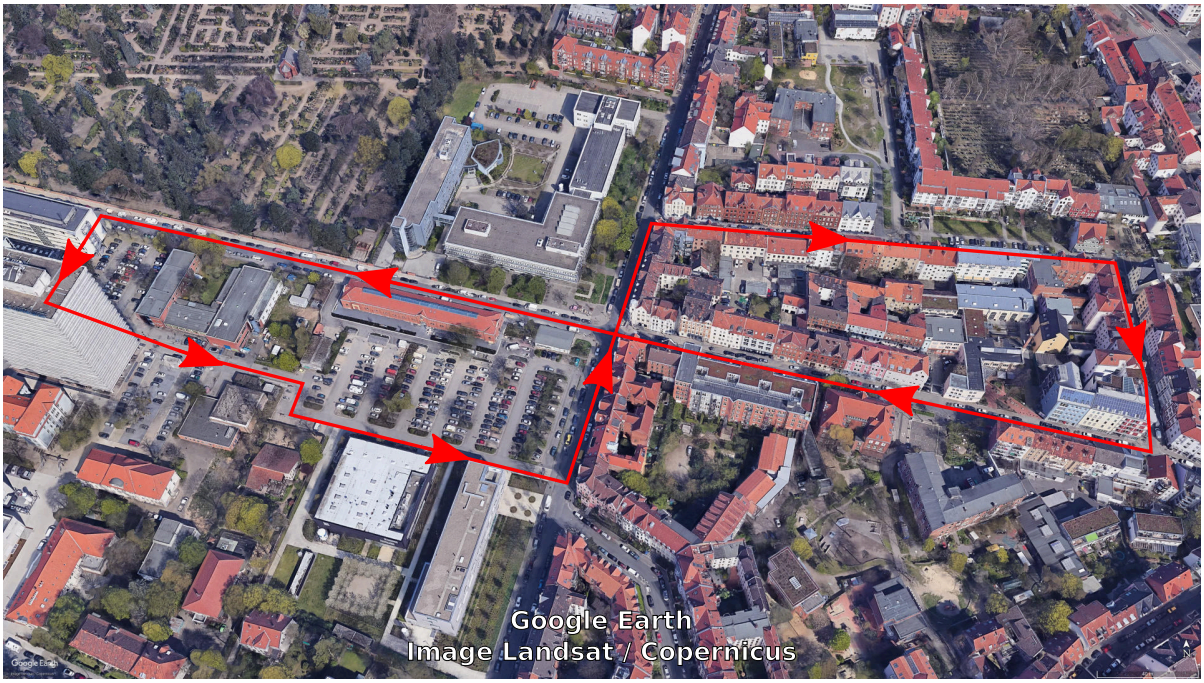


Figure 8.13: Overview of the test environment with the route drawn in red. Source: Google Earth, Image credit: Landsat / Copernicus.

Naturally, the error bounds differ from sensor to sensor. Thus, in the following we derive or determine these bounds for the sensors introduced in the previous section that are then employed for all following experiments.

Laser scanner

According to Section 4.2.1, we need to determine the uncertainties of the distance measurement $[\Delta_r]$, the vertical angle $[\Delta_\theta]$, the horizontal angle $[\Delta_\varphi]$ and the dimensions of the initial footprint $[\Delta_{b_\theta}]$ and $[\Delta_{b_\varphi}]$.

Regarding the distance uncertainty $[\Delta_r]$, the manufacturer specifies an accuracy of $\pm 2\,\text{cm}$ (one sigma). Consequently, to bound the distance measurements with a probability of $99.7\,\%$, we choose $[\Delta_r] = [-6, 6]\,\text{cm}$. Next, the manufacturer specifies the beam divergence of each laser beam as 3 mrad. Unfortunately, there is no distinction between the horizontal and the vertical beam divergence, and thus we set $[\Delta_\theta] = [\Delta_\varphi] = [-1.5, 1.5]\,\text{mrad}$. Last, we need to determine the initial footprint of each laser beam, i.e. its dimensions when it is emitted. However, it is not specified by the manufacturer, and there are no experiments aimed at determining this initial footprint. We therefore assume the laser emitters of the Velodyne HDL-64E to be similar to those of the VLP-16 mentioned above. Consequently, we choose $[\Delta_{b_\theta}] = [-4.75, 4.75]\,\text{mm}$ and $[\Delta_{b_\varphi}] = [-6.35, 6.35]\,\text{mm}$.

Camera

According to Section 4.2.2, we need to determine intervals enclosing the feature detection error caused by quantization $[\Delta_{px_q}]$, by image blur $[\Delta_{px_b}]$ and by measurement noise $[\Delta_{px_n}]$.

Naturally, $[\Delta_{px_q}] = [-0.5, 0.5]$ px since the actual scene is discretized into pixels. In contrast, image blur and measurement noise cannot be quantified just as easily since many different factors - some of which are unknown - have an impact. Consequently, we can only choose error bounds $[\Delta_{px}]$ that summarize all of the previously mentioned types of error. However, to our knowledge there exists no studies on the error distribution of image feature matches. Thus, we need to select $[\Delta_{px}]$ empirically and rather conservatively to not underestimate the error. For our experiments, we found that error bounds that scale with the distance from the last keyframe are superior to fixed error bounds. The reason is that the accuracy of the feature matches decreases the further away from each other the images have been recorded. Consequently, $[\Delta_{px}](t) = [\Delta_{px_f}] + d(t) \cdot [\Delta_{px_s}]$, where $d(t)$ is the distance from the last keyframe at time t and $[\Delta_{px_f}]$ is an error bound that is fixed for every image frame and $[\Delta_{px_s}]$ is an error bound that scales with the distance from the last keyframe.

We set the error bounds $[\Delta_{px_f}] = [-2, 2]$ px and $[\Delta_{px_s}] = [-0.1, 0.1]$ px that are guaranteed to enclose most image feature matches, while some are obvious outliers and need to be treated using a relaxed intersection, as explained in Section 7.2.4.

IMU

The derivation of the sensor error bounds for the IMU that we employ in this experiment can be found in Section 8.1.2.2, since we employ the same sensor to measure the same data as in this section.

8.3.3 Parameters

Before finally presenting results of our approach for guaranteed visual-LiDAR odometry, we recall some important parameters that were introduced in Chapter 7.

First, we limit the number of feature matches for every image pair. To do so while simultaneously ensuring a proper distribution of image features, we divide each image into 20×15 sub-windows. For the first results, we allow a maximum of 3 image features in each of this 300 sub-windows. However, in Section 8.3.7 we vary the number of features and analyze the effect of this parameter.

Second, we have to choose a maximum size of the interval vector enclosing the pose parameters before inserting a new keyframe as introduced in Section 7.2.3. Here, we only monitor the area of the position box projected onto the ground plane. Other possibilities for determining the uncertainty of the pose estimate include the volume of the position box or the interval width of the Euler angles. However, we discovered that it is sufficient to consider the area of the position box since it is directly related to the volume of the position box and the rotation uncertainty. Thus, if the area increases, the volume and the rotation uncertainty increase simultaneously. For the first results, we insert a new keyframe once the area of the position box exceeds $5\,\mathrm{m}^2$, but we also vary and analyze this parameter more closely in Section 8.3.8.

Third, we have to specify the maximum number of outliers to expect for the relaxed intersection explained in Section 7.2.4. We choose a maximum of $5\,\%$ outliers that we

determined empirically and selected rather conservatively to keep the guarantees intact. However, we will also vary this parameter and inspect it more closely in Section 8.3.9.

Next, we specify the spatiotemporal uncertainties used to fuse information from camera, laser scanner and IMU. As stated in Section 8.3.1, we employed an external laser tracker to accurately measure the true extrinsic calibration between our sensors. Moreover, we use GPS timestamps to ensure an adequate time synchronization of sensor data streams. Although we could have used our new approaches for the spatiotemporal calibration of sensors, we decided to decouple the experiments. This allows us to evaluate the bounded-error visual-LiDAR odometry approach using different accuracies for the spatiotemporal calibration parameters while simultaneously ensuring that the true parameters are enclosed. Nevertheless, for the first results we employ error bounds that are similar to the uncertainties discovered during the experimental evaluation of the calibration approaches in Section 8.1 and Section 8.2.

First, we focus on the extrinsic calibration between camera and LiDAR. For the translation parameters, we choose an uncertainty of $[-5, 5]$ cm for $[{}_xT_L^C]$ and $[{}_yT_L^C]$, and an uncertainty of $[-1.5, 1.5]$ cm for $[{}_zT_L^C]$. For the rotation parameters, we choose an uncertainty of $[-0.35, 0.35]^\circ$ for $[\phi_L^C]$ and $[\theta_L^C]$, and an uncertainty of $[-0.2, 0.2]^\circ$ for $[\psi_L^C]$. All of these uncertainties should result in the transformation parameter intervals enclosing the true parameters and are chosen even more conservatively than the results in Section 8.2.2 to not underestimate the possible error.

Second, we focus on the spatiotemporal calibration between camera and IMU. Unfortunately, the extrinsic rotation accuracy depicted in Section 8.1 is too low. Using the interval bounds shown there for sensor fusion would not make it possible to provide an initial rotation estimate using IMU data. In contrast, our experiments here show that an uncertainty of $[-0.5, 0.5]^\circ$ that we add to all three rows of $[\boldsymbol{\xi}_C^I]$ is required to provide a reasonably accurate guess for the rotation of the vehicle. For the time offset, we choose an uncertainty of $[\tau] = [-7.5, 7.5]$ ms. This is in accordance with the results in Section 8.1.

Finally, we found that coupling the forward-backward contractors with SIVIA as explained in Section 7.3 does not significantly increase the accuracy of our pose estimates and therefore does not justify the additional computation time.

8.3.4 Comparable state-of-the-art approach

To compare our results to those of a state-of-the-art approach, we implemented the frame to frame motion estimation by Zhang et al. [60, 61]. Similar to our work, the authors of this work project the laser scan point cloud to the image plane and are therefore able to compute the depth of the image features. However, they only consider the three closest scan points for each image feature and compute the depth by interpolating between those three points. Consequently, they are not able to assess the accuracy of the fused 3D features. Afterwards, the authors employ a nonlinear optimization algorithm (in this case the Levenberg-Marquardt algorithm [14]) to compute the rigid body transformation corresponding to the robot's pose. Here, they employ the same equations we depict in Section 7.2.2. Besides the frame to frame motion estimation, the authors' approach also incorporates an approach for bundle adjustment. To keep the comparison fair, however, we only implemented the frame to frame motion estimation since our own approach does not perform bundle adjustment. Moreover, we

feed the very same image features, laser scan points and initial rotation estimate by the IMU to both our and this stochastic approach.

8.3.5 Results

Finally, this section details the results of our interval-based visual-LiDAR odometry approach using the sensor error bounds and parameters depicted previously. Moreover, we provide a comparison with the state-of-the-art approach introduced in the previous section.

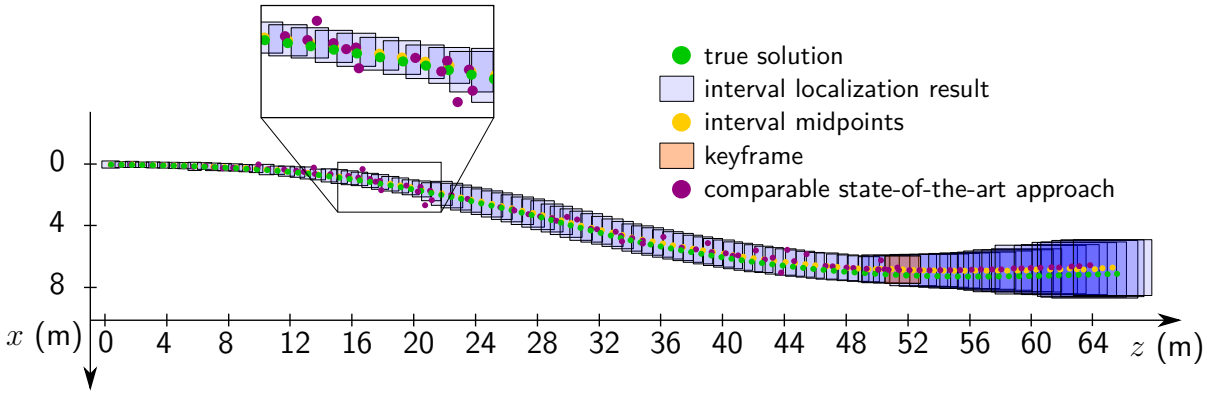


Figure 8.14: Short experiment (10 s) depicting the increasing uncertainty due to error propagation at keyframes. Green dots depict the true solution, blue boxes illustrate our interval localization results, yellow dots depict their respective midpoints, orange boxes highlight keyframes and violet dots depict the result of the comparable state-of-the-art approach.

Since our approach only serves the purpose of computing the robot's pose relative to the last keyframe, we only evaluate this relative, but no global localization results. Nevertheless, it is possible, to combine the transformations between multiple keyframes as explained in Fig. 7.11 to compute the robot's motion not only between keyframes, but also over a longer period of time. However, this results in a rapidly growing uncertainty since the uncertainty that accumulates until the insertion of a new keyframe cannot be contracted in the future. An example of this behavior, which is typical for dead-reckoning approaches, can be seen in Fig. 8.14. Here, we use the first 10 s of our dataset. It becomes evident that all boxes contain the true solution and the interval midpoints approximate the true result reasonably well. However, after a few seconds, the uncertainty grows too large to provide any meaningful bounds on the vehicle's pose. After 10 s, the last position box of this small experiment covers an area of 18 m^2 and has a volume of 52 m^3. Nevertheless, this uncertainty allows us to warn the user or determine when external information is needed.

In contrast, the stochastic approach also drifts from the true result and has an error of 1.81 m on the ground plane and an error of 1.84 m in 3D after 10 s. However, the stochastic approach provides no means to assess the uncertainty of its position estimates. As a further comparison, the midpoint of the interval box has an error of 0.49 m on the ground plane and an error of 0.55 m in 3D. In addition, as can be seen, some stochastic pose estimates are outside the computed boxes, and so we can be sure that the stochastic result must be wrong in these cases. A possible explanation of why the stochastic approach fails in some cases while our

interval-based approach is still able to compute a box containing the true solution will be given later.

The findings of this first small experiment are that global contractors should be developed in the future to prevent the uncertainty from increasing rapidly. These global contractors can be found, for example, by extending our approach by bundle adjustment, extending it to SLAM, or by using GNSS information.

To evaluate the performance of our approach, we define the following metrics:

- **Correct** (%): The percentage of image frames for which the true pose is enclosed in the computed pose intervals.
- **Volume** (m^3): The average volume of our 3D boxes enclosing the robot's true position.
- **Area** (m^2): The average area of our 3D position boxes that are projected onto the ground plane.
- **Rotation accuracy** (°): The average width of the interval enclosing the robot's orientation on the ground plane.
- **Features w. depth**: The average number of image feature matches for which we were able to find depth information.
- **Distance per keyframe** (m): The average driving distance after which we insert a new keyframe due to the position uncertainty becoming too large.
- **Inconsistencies** (%): The percentage of image frames for which the result of the state-of-the-art approach [60, 61] is not enclosed in the 6 DOF pose intervals.

Correct (%)	Volume (m^3)	Area (m^2)	Rotation accuracy (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
100	0.77	1.30	0.77	105	11.35	29.9

Table 8.23: Quantitative results of the interval-based visual-LiDAR odometry approach.

Table 8.23 depicts the results. As can be seen, we are able to reliably enclose the robot's true pose in all 2644 image frames, and are thus able to provide the desired guarantees. Moreover, the accuracy of our approach is reasonable, considering that we take the maximum error into account and allow up to 5 % outliers. Despite this accuracy, we are able to detect an inconsistency of the state-of-the-art approach in 29.9 % of all image frames, showing that our approach can be employed to alert the user in the case of errors.

In addition, Fig. 8.15 shows the originally three-dimensional position boxes that are projected onto the ground plane. Since we only compute the robot's pose relative to the last keyframe, we use ground truth information at every keyframe to transform the position boxes into a global coordinate system for this visualization. As can be seen, the size and thus the uncertainty of the position boxes as well as the frequency of keyframes vary for different parts of the trajectory. To explain this phenomenon, we present the image features of two different points in our experiment on the bottom of Fig. 8.15. It becomes evident that the image on the right is recorded in an environment, which makes it easy to re-identify image features and to assign accurate depth information to image features on planes parallel to the image plane (e.g. on the cars). In contrast, the image on the left is captured in an environment with less

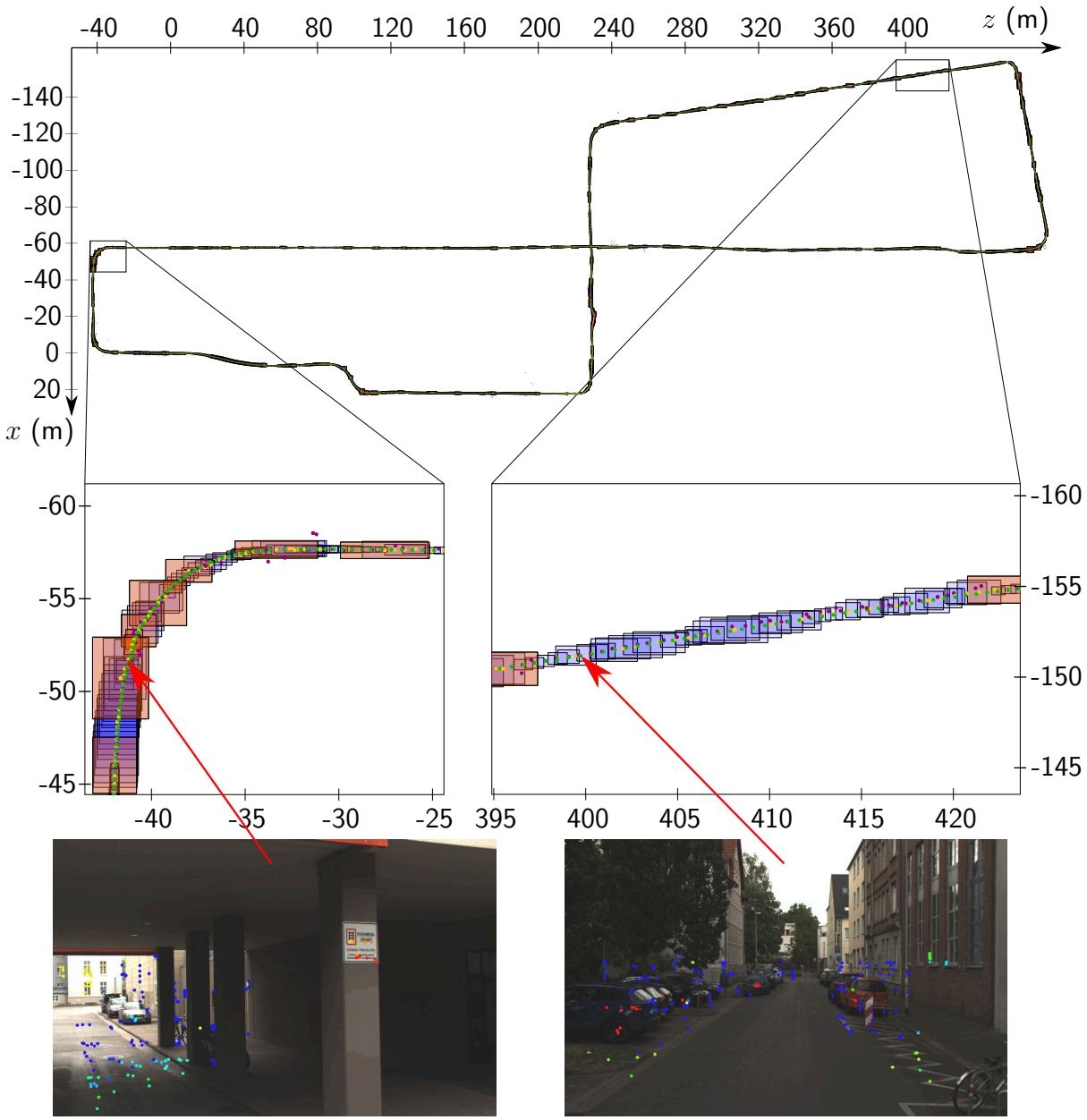


Figure 8.15: Visualization of the boxes enclosing the robot's position in the x-z-plane. Green dots depict the true solution, blue boxes illustrate our interval localization results, orange boxes highlight keyframes and violet dots depict the solution of the stochastic approach. All results are transformed into a global coordinate system using ground truth information at every keyframe. In addition, on the bottom left and right we show feature images that are typical for the corresponding sections of the trajectory. Here, the image features are colored coded by depth uncertainty with red signaling accurate features and blue signaling inaccurate features.

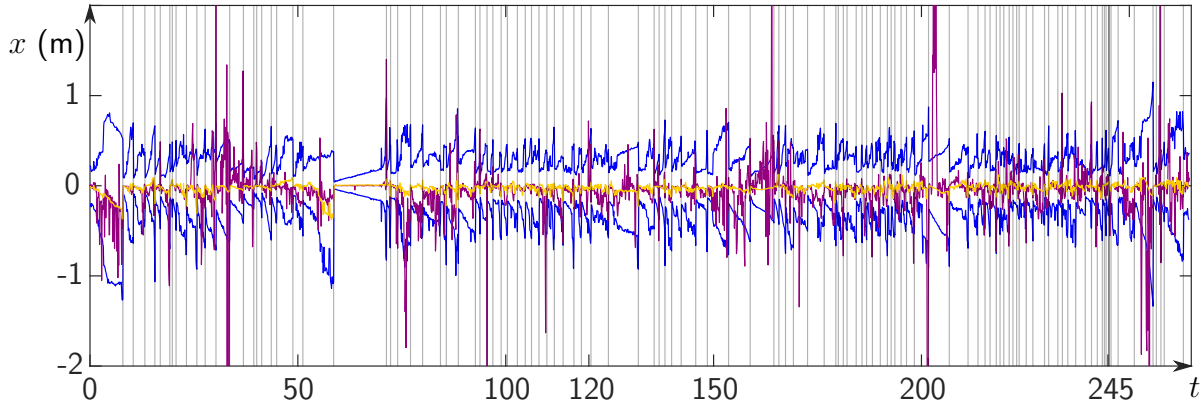
rich image features. Moreover, there are fewer features with accurate depth information due to the discontinuity of the environment (e.g. caused by the pillars). Consequently, the constraints introduced by the 3D features of the right image are stronger than those introduced by the 3D features of the left image, thus resulting in a more accurate contraction of the robot's pose at the time of the right image than at the time of the left image.

Furthermore, Fig. 8.16 shows the interval pose bounds and the results of the stochastic approach with respect to ground truth information. Since the car moves mainly in two dimensions, we only show results for the translation in x- and z-direction and the rotation around the y-axis. It is evident that our error bounds never violate the true result (i.e. 0 is always enclosed in those intervals), and thus we can state that our intervals are guaranteed to contain the true solution. Furthermore, it can be seen that the midpoint of our intervals approximates the true result reasonably well, and thus it can be used if a point-valued result is needed. The next characteristic we want to point out is that the width of our intervals, and thus the uncertainty of the localization results, varies drastically. This can also be seen by the fact that keyframes are inserted more frequently for some parts of the trajectory.

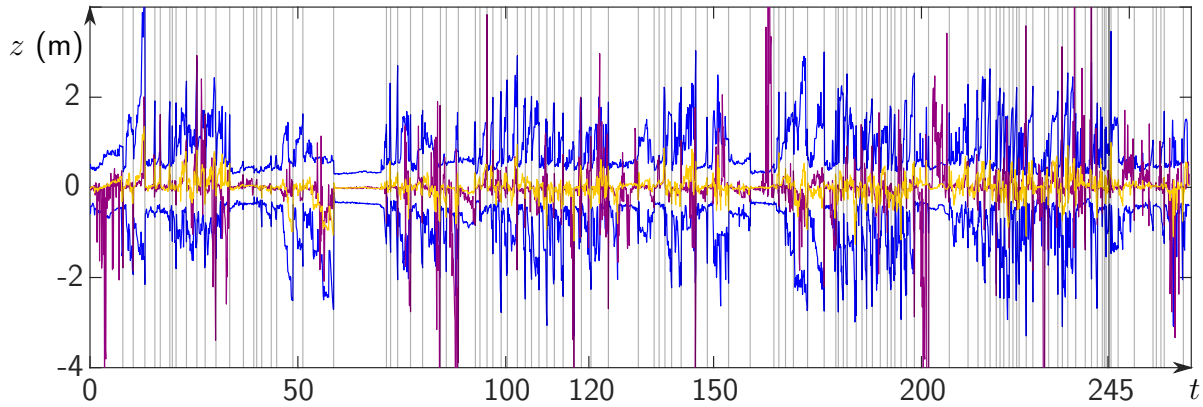
To further illustrate this behavior and provide more details, Fig. 8.17 shows two smaller sections of the experiment. As can be seen, significantly less keyframes are inserted between 100 s and 150 s (cf. Fig. 8.17b, Fig. 8.17d and Fig. 8.17f). In contrast, more keyframes keyframes are inserted between 210 s and 260 s (cf. Fig. 8.17a, Fig. 8.17c and Fig. 8.17e). This is due to the fact that different areas of the trajectory provide 3D features of different numbers and different uncertainties. For example, the right image of Fig. 8.15 is recorded around $t = 120\,\text{s}$, whereas the left image of Fig. 8.15 is captured around $t = 245\,\text{s}$.

Moreover, these figures illustrate the inconsistencies between the results of the stochastic and our interval-based approach. Whenever the result of the stochastic approach lies outside the lower and upper interval bounds (i.e. the red line crosses a blue line), the stochastic solution cannot be correct and therefore an inconsistency has occurred. We believe that these inconsistencies occur due to unmodeled uncertainties of the 3D features. Since the stochastic approach disregards any uncertainty the augmented features might have, better features cannot be assigned a higher weight, and thus many bad features overrule the few good features. In contrast, our interval-based approach considers the uncertainties during sensor fusion. Features for which depth cannot be determined accurately (e.g. on borders) are assigned large intervals and vice versa. Consequently, the constraints imposed by uncertain features do not contribute much to the contraction process. In contrast, features for which depth could be determined accurately impose stronger constraints on the rigid body transformation.

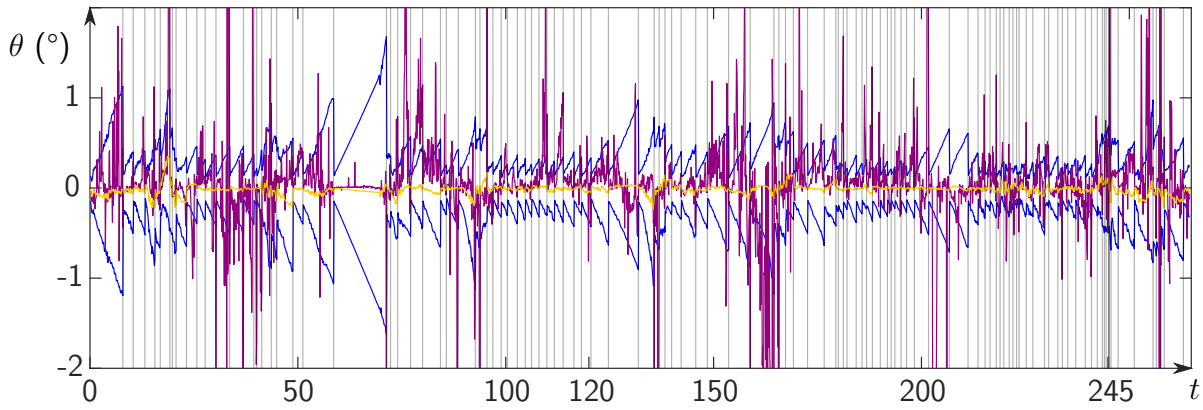
Fig. 8.18 shows augmented feature images for both our and the stochastic approach during the detected faults (especially for the translation along the z-axis) around $t = 200\,\text{s}$. First, it can be seen that our approach uses some features without depth, for which the stochastic approach finds depth. This is due to the fact that our approach is able to dynamically check if the image feature interval is completely covered by scan point intervals and does not assign depth information if this is not the case. In contrast, the conventional approach only requires three scan points to be close enough to the image feature. Except for these few features, it can be seen that the depth estimates by the stochastic approach (cf. Fig. 8.18a) look similar to the mid points of our depth intervals (cf. Fig. 8.18b). However, the uncertainty of our depth



(a) Error of the translation along the x-axis.



(b) Error of the translation along the z-axis.



(c) Error of the rotation around the y-axis.

Figure 8.16: Visualization of the results of our interval-based visual-LiDAR odometry approach. The three graphs show the relative error (i.e. ground truth is 0) over time. Each vertical gray line corresponds to the insertion of a new keyframe (thus, position estimates are always relative to the last vertical line). The lower and upper bounds of our method are depicted in blue, the corresponding midpoints are colored yellow and the result of the stochastic method is colored violet. More detailed sections of this graphs are provided in Fig. 8.17.

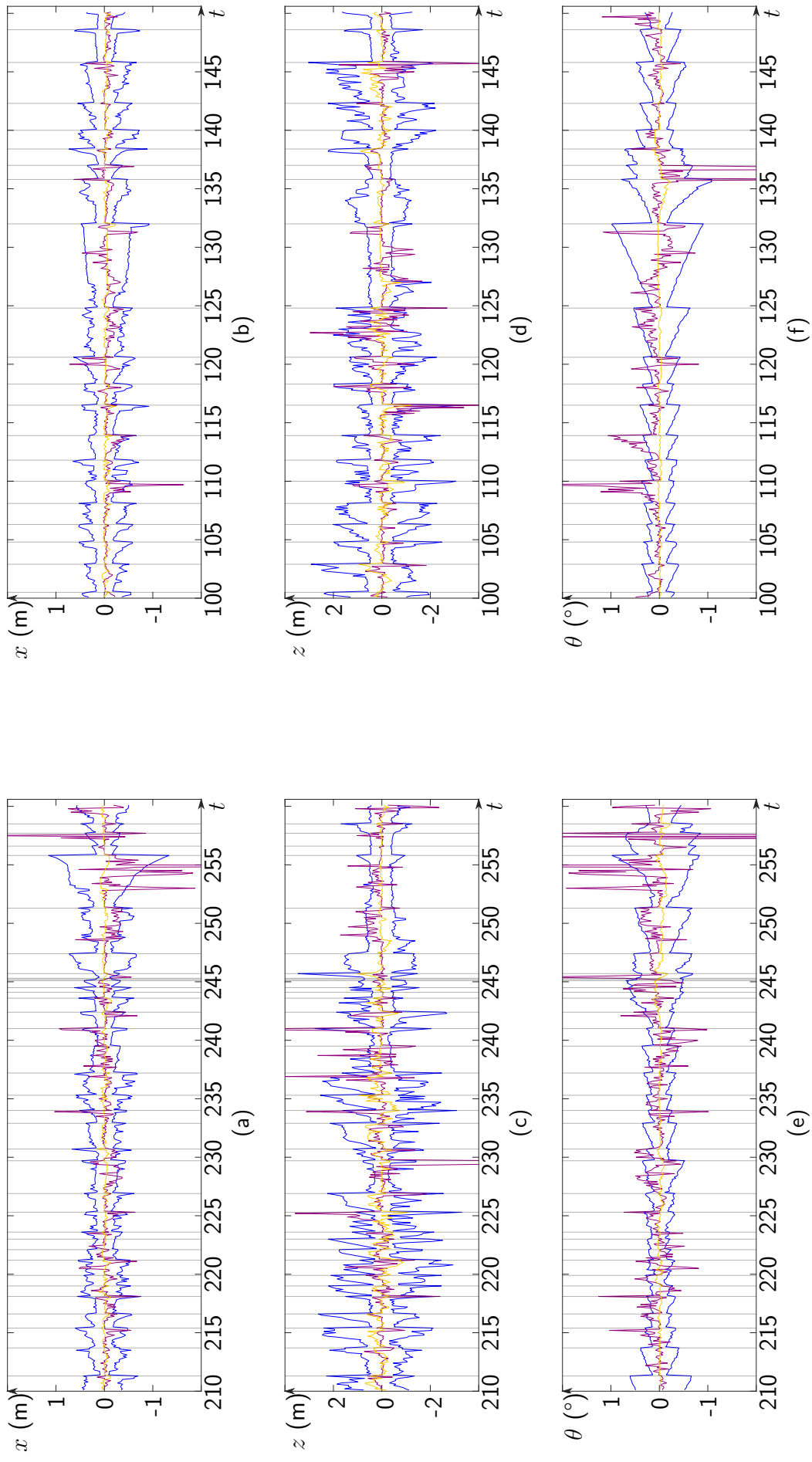


Figure 8.17: Visualization of the relative error for two smaller sections of the experiment (cf. Fig. 8.16). The graphs on the left correspond to the left and the graphs on the right correspond to the right image of Fig. 8.15.

(a) Stochastic features colored by depth (red: close, blue: distant, violet: no depth).

(b) Interval features colored by depth midpoint (red: close, blue: distant, violet: no depth).

(c) Interval features colored by uncertainty (red: certain, blue: uncertain).

Figure 8.18: Image features for the detected fault around $t = 200\,\mathrm{s}$.

intervals indicates that only a few features are accurate and should be trusted (cf. Fig. 8.18c). Although stochastic approaches successfully apply various methods to cope with erroneous position estimates (e.g. Zhang et al. employ bundle adjustment to smooth the computed trajectory), we still believe that detecting these faults in a guaranteed way, allows stochastic approaches to be more robust to unfavorable features.

Finally, we mention and analyze the computation time of our approach. On average, the computations for one image frame take 0.91 s. This computation time can be divided into 0.26 s for detecting and tracking image features, 0.13 s for assigning depth information to image features, 0.02 s for determining outliers, 0.41 s for building the forward-backward contractors for every image feature match and 0.09 s for running the aforementioned contractors to contract the pose domains. These computation times were achieved on a consumer-grade laptop with little effort for threading or code optimizations making us optimistic that this approach is suitable for future online applications. Parallelization in particular can be used to speed up the computations. Moreover, while the IBEX library allows a simple implementation of contractors, it was developed without special consideration of computation time, and thus the construction of forward-backward contractors takes a considerable amount of time. However, the execution of those contractors is comparably fast and we are therefore optimistic for a future online implementation.

8.3.6 Error bounds

After presenting the general results, we conduct a parameter study starting with different error bounds for the image pixel error of the camera. In the following, we refrain from varying the sensor error bounds of the laser scanner and the IMU since these are thoroughly described in the data sheets, and thus assumed to be correct.

Camera

At first, we vary only the fixed image pixel error $[\Delta_{px_f}]$ while retaining all other parameters and error bounds as described above.

$r([\Delta_{px_f}])$ (px)	Correct (%)	Volume (m^3)	Area (m^2)	Rotation error (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
0.5	99.47	0.60	1.15	0.80	94	11.86	25.9
1.0	99.77	0.67	1.18	0.83	92	12.30	27.5
1.5	99.96	0.70	1.25	0.80	98	11.97	29.9
2.0	100	0.77	1.30	0.77	105	11.35	29.9
2.5	100	0.77	1.35	0.76	108	11.16	25.2
3.0	100	0.74	1.35	0.75	109	10.98	25.3
3.5	100	0.85	1.44	0.74	107	10.80	24.2
4.0	100	0.95	1.52	0.75	111	10.46	22.0

Table 8.24: Results of the parameter study on the fixed image pixel error $[\Delta_{px_f}]$.

The results can be seen in Table 8.24. As expected, if $r([\Delta_{px_f}])$ is chosen too small, we are no longer able to enclose the correct solution for every image frame, and thus the results are not guaranteed. However, our approach is still able to compute a correct solution for more than 99 % of all image frames. In contrast, if $r([\Delta_{px_f}])$ is chosen too large, the accuracy of our results decreases as can be seen by the fact that the average area and the average volume increases. Besides, more keyframes have to be inserted. This is also in accordance with our expectations since large error bounds that are never reached by the sensor dilute the results (cf. Section 3.11). Finally, we want to mention that the variations of the number of image features with depth can be explained by the fact that the image pixel error is also employed to filter outliers during the image feature matching process (cf. Section 7.2.4).

Time offset between camera and IMU

Next, we vary the accuracy of the time offset interval $[\tau]$. Since the sensors are synchronized using GPS timestamps, we assume that $\tau^* = 0$. However, an error, which we cannot assess, may still remain. All other parameters and sensor error bounds are retained.

Table 8.25 shows the results. As can be seen, assuming no uncertainty (i.e. $r([\tau]) = 0.0$ ms) results in some erroneous pose estimates due to the initial rotation intervals not containing

$r([\tau])$ (ms)	Correct (%)	Volume (m^3)	Area (m^2)	Rotation error (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
0.0	99.77	0.69	1.23	0.70	97	11.86	29.4
7.5	100	0.77	1.30	0.77	105	11.35	29.9
15.0	100	0.80	1.34	0.88	105	11.35	25.2
30.0	100	0.85	1.38	1.08	105	11.55	23.8
50.0	100	0.88	1.42	1.28	106	10.80	20.7

Table 8.25: Results of the parameter study on the time offset between camera and IMU $[\tau]$.

the true rotation. This can either be due to the extrinsic rotation uncertainty being set too optimistically or due to an imperfect synchronization of the sensors.

Naturally, a more uncertain time offset results in less accurate rotation estimates. Since the accuracy of the translation parameters depends not only on the accuracy of the 3D features but also on the accuracy of the rotation, the position estimate also becomes less accurate with an increasing uncertainty of the time offset. In turn, an increasing uncertainty of the position estimates leads to a more frequent insertion of keyframes, since the threshold for the maximum area of a position box is exceeded more often. Moreover, the number of feature matches with depth increases with the uncertainty of the time offset. The reason is the outlier detection (cf. Section 7.2.4) that finds feature matches that are not consistent with the initial rotation estimate by the IMU.

Extrinsic rotation between camera and IMU

$r([\boldsymbol{\xi}_C^I])$ (°)	Correct (%)	Volume (m^3)	Area (m^2)	Rotation error (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
0.1	97.96	0.59	1.20	0.59	95	12.18	41.4
0.25	99.51	0.62	1.21	0.67	98	11.96	35.8
0.5	100	0.77	1.30	0.77	105	11.35	29.9
0.75	100	0.86	1.38	0.90	107	11.35	23.0
1.0	100	0.96	1.43	1.02	108	11.07	19.1
2.0	100	1.51	1.66	1.60	105	10.30	14.8

Table 8.26: Results of the parameter study on the extrinsic rotation between camera and IMU $[\mathbf{R}_C^I]$.

In the following, we evaluate the influence of the uncertainty of the extrinsic rotation between camera and IMU. Initially, we set the uncertainty of all three Euler angles $[\boldsymbol{\xi}_C^I]$ to 0.5° and did not distinguish between the three different rotation axes. Here, we maintain this procedure and adapt the accuracy of all three Euler angles simultaneously. The intervals for the Euler

angles should be centered on the true rotation parameters since we calibrated the sensor setup externally as explained above. However, we cannot assess the accuracy of this calibration, and thus an error may remain such that setting $r([\boldsymbol{\xi}_C^I]) = 0°$ is infeasible.

Table 8.26 depicts the results. The interpretation is similar to the previous paragraph, since an increasing uncertainty of the extrinsic rotation - similar to an increasing uncertainty of the time offset - leads to less accurate rotation estimates by the IMU.

Extrinsic transformation between camera and laser scanner

Next, we focus on the accuracy of the extrinsic transformation between camera and laser scanner. As for the extrinsic calibration between camera and IMU, we used an external calibration system to accurately compute the 6 DOF transformation. Afterwards, we added an uncertainty to each parameter that corresponds to the accuracy our interval-based calibration approach achieves. Consequently, the transformation intervals $[\mathbf{T}_L^C] = (\,[{}_xT_L^C]\ \ [{}_xT_L^C]\ \ [{}_xT_L^C]\,)$ and $[\boldsymbol{\xi}_L^C] = (\,[\phi_L^C]\ \ [\theta_L^C]\ \ [\psi_L^C]\,)$ should be centered on the true values. Even for the external calibration system, however, there remains a certain degree of inaccuracy.

Trial	$r([{}_xT_L^C])$ (cm)	$r([{}_xT_L^C])$ (cm)	$r([{}_xT_L^C])$ (cm)	$r([\phi_L^C])$ (°)	$r([\theta_L^C])$ (°)	$r([\psi_L^C])$ (°)
1	3	3	1.0	0.20	0.20	0.1
2	5	5	1.5	0.35	0.35	0.2
3	7	7	2.0	0.50	0.50	0.3
4	3	3	1.0	0.35	0.35	0.2
5	5	5	1.5	0.20	0.20	0.1

Table 8.27: Listing of the trial number and the corresponding extrinsic transformation uncertainties between camera and laser scanner.

Since the inaccuracies of all six parameters are set differently, we cannot fit the parameter inaccuracies and corresponding results into a common table. Consequently, Table 8.27 assigns a trial number to each combination of parameter uncertainties.

Finally, Table 8.28 depicts the results. As can be seen, we are able to reliably enclose the correct solution in all five experiments, which shows that that the external extrinsic calibration parameters are more accurate than initially assumed. Naturally, more accurate transformation parameters result in a more accurate contraction of the robot's pose as can be seen by the average area and average volume for the first, fourth and fifth trial. In contrast, less accurate transformation parameters for the third trial lead to less accurate pose contractions. Consequently, this experiment shows the importance of the accuracy of the extrinsic calibration between camera and LiDAR. Therefore, our extrinsic calibration approach should be further improved in the future to enable more accurate visual-LiDAR odometry.

Another effect of more accurate transformation parameters is that fewer keyframes have to be inserted. The reason for this is directly linked to the more accurate contraction of the robot's pose. Since we insert a new keyframe whenever the position box exceeds a maximum

Trial	Correct (%)	Volume (m^3)	Area (m^2)	Rotation error (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
1	100	0.54	0.94	0.79	88	12.65	31.5
2	100	0.77	1.30	0.77	105	11.35	29.9
3	100	0.94	1.77	0.75	110	8.91	22.0
4	100	0.73	1.19	0.79	101	12.18	26.7
5	100	0.59	1.04	0.80	92	12.53	28.1

Table 8.28: Results of the parameter study on the extrinsic transformation between camera and laser scanner consisting of the rotation $[\mathbf{R}_L^C]$ and the translation $[\mathbf{T}_L^C]$.

threshold, but this threshold is reached less often with more accurate pose parameters, fewer keyframes have to be inserted. Consequently, fewer features with depths are found if the transformation parameters are more accurate. The reason for this is that more image feature matches are found immediately after a new keyframe is inserted, and the further we are from the last keyframe, the fewer matches with image features remain.

Naturally, we are also able to detect more inconsistencies using more accurate transformation parameters. The reason is twofold. As explained, more accurate transformation parameters result in a more accurate contraction of the pose parameters, and thus even smaller deviations of the stochastic approach can be identified. Second, the stochastic approach performs worse the further we are from the last keyframe since fewer features remain, and thus even few outliers or features with inaccurate depth information significantly distort the result. Consequently, weighting those features becomes even more important.

Another finding of these experiments is that, as expected, the rotational uncertainty has a greater impact on the final accuracy, as a comparison of the results for the fourth and fifth trials shows.

8.3.7 Number of features

Next, we focus on the number of features we select in each of the 20×15 sub-windows. Again, all other parameters and sensor error bounds are retained as described above.

Table 8.29 shows the results. As can be seen, only with at least two features per window, our approach is guaranteed to enclose the correct solution. However, if we only select one feature per window, our approach fails for two image frames and does not compute a box containing the true solution. This can be explained as follows. If we use only a few features for the contraction but compute the maximum number of outliers proportionally, sometimes even one or two outliers violate this maximum number of outliers. Consequently, we require a sufficient number of features to permit the proportional computation of expected outliers.

Moreover, the uncertainty - as indicated by the average volume and average - increases with the number of features per window. At first glance, this appears counterintuitive, since established approaches can usually find more accurate solutions the more information they obtain. This can also be seen by the fact that the percentage of inconsistencies decreases if

Number of features	Correct (%)	Volume (m^3)	Area (m^2)	Rotation error (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
1	99.92	0.62	1.21	0.74	45	11.26	43.9
2	100	0.67	1.25	0.77	76	11.75	33.7
3	100	0.77	1.30	0.77	105	11.35	29.9
4	100	0.77	1.33	0.76	124	11.45	24.7
5	100	0.77	1.34	0.76	139	11.26	23.3
6	100	0.78	1.37	0.75	158	11.45	21.9

Table 8.29: Results of the parameter study on the number of features we select in each of the 20×15 sub-windows.

we allow more features per window. However, for our approach it can be seen that adding as much information as possible (i.e. allowing up to six features per window) is counter-productive. This is because the additional features do not always add new information (e.g. if two features are close by and have the same 3D coordinates), but increase the overall number of features, which in turn increases the maximum number of outliers that is computed proportionally. Consequently, the result is diluted. In summary, we conclude that our approach can compute a guaranteed solution even if few image features are provided while this is not the case for the stochastic approach that fails for almost every third image frame.

Finally, allowing more features directly increases the computation time of our approach since additional contractors must be built and executed. While the average computation time per image frame using one feature per window is 0.6 s, it increases to 1.3 s if we use six features per window.

8.3.8 Keyframe insertion criteria

Next, we vary the maximum area the position box must not exceed before a new keyframe is inserted. As previously explained, the position box is projected to the ground plane to compute its area. Again, all other parameters and sensor error bounds are retained.

The results can be seen in Table 8.30. Again, our approach provides the desired guarantees and encloses 100 % of the true solutions. Naturally, if the keyframe insertion criterion is more strict (i.e. the maximum area of a position box is 2.5 m^2), more keyframes are inserted, and thus the average distance per keyframe decreases. Simultaneously, the uncertainty of the pose estimates - as indicated by the average area, the average volume and the average rotation error - decreases. Moreover, the average number of features with depth increases. This is expected since we find more features directly after inserting a keyframe. Thus, inserting a keyframe more often results in more feature matches.

Finally, the number of inconsistencies is the lowest for a maximum area of 2.5 m^2. This can be explained as follows. Our interval-based approach is capable of computing an - albeit inaccurate - pose box containing the true solution even for few feature matches due to its ability to take the uncertainty of the depth information into account. Thus, if we allow a larger

Maximum area (m^2)	Correct (%)	Volume (m^3)	Area (m^2)	Rotation error (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
2.5	100	0.39	0.88	0.70	115	8.62	22.7
5	100	0.77	1.30	0.77	105	11.35	29.9
7.5	100	1.08	1.68	0.83	94	13.15	27.6
10	100	1.27	1.91	0.81	93	13.69	24.5
15	100	2.12	2.71	0.86	88	14.92	28.8

Table 8.30: Results of the parameter study on the maximum area of our position box on the ground plane before we insert a new keyframe.

maximum area before inserting a new keyframe, our interval-based approach can still compute a box containing the true solution even if the robot moved a considerable distance from the last keyframe and only a few image feature matches remain. In contrast, the stochastic approach cannot distinguish between accurate and inaccurate 3D feature matches, and thus fails to converge to the correct solution if few feature matches are provided. Consequently, since more keyframes are inserted and more feature matches are found for the most strict keyframe insertion criterion, the stochastic approach is less likely to fail for a maximum area of 2.5 m^2.

8.3.9 Outlier percentage

In this section, we vary the maximum number of outliers to expect for the relaxed intersection. The remaining parameters and sensor error bounds are retained.

Maximum outliers (%)	Correct (%)	Volume (m^3)	Area (m^2)	Rotation error (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
10	100	0.82	1.66	0.72	115	9.42	20.8
9	100	0.89	1.64	0.74	111	9.77	21.7
8	100	0.86	1.56	0.75	109	10.21	22.4
7	100	0.86	1.46	0.77	105	10.71	26.5
6	100	0.80	1.40	0.77	105	11.35	27.3
5	100	0.77	1.30	0.77	105	11.35	29.9
4	99.85	0.72	1.24	0.81	95	12.18	27.0
3	99.81	0.71	1.16	0.81	94	12.53	27.2
2	99.81	0.67	1.07	0.82	94	12.89	31.6
1	99.66	0.62	0.94	0.83	93	13.15	29.7
0	96.86	0.48	0.73	0.73	105	10.71	32.8

Table 8.31: Results of the parameter study on the maximum number of outliers.

Table 8.31 shows the results. It is evident that a decreasing maximum outlier percentage results in an increasing accuracy of our pose estimates. In addition, fewer keyframes have to be inserted. However, if the maximum outlier percentage is chosen too large, not all pose boxes contain the true solution, and thus the results are no longer guaranteed. This again underlines the importance of choosing interval bounds as narrow as possible but as large as necessary (cf. Section 3.11).

Decreasing the maximum outlier percentage to 0 reverses the downward trend of features with depth information and the upward trend of the average distance per keyframe. This can be explained as follows. Whenever the pose contraction results in an empty box, we have to insert a new keyframe to be able to compute the current pose. This is exactly what happens when setting the maximum outlier percentage to 0, because in this case even a single outlier can lead to an empty pose box. As a result, more keyframes are inserted, and the number of features increases again since more features are found immediately after a new keyframe is set.

Nevertheless, it can be seen that even with the assumption of all feature matches being correct, only $3.14\,\%$ of all 2644 pose boxes do not contain the correct solution. Consequently, we conclude that mismatched features only occur for few images.

8.3.10 Without outliers

Finally, we want to highlight the potential of our approach if the feature matching would be flawless. Therefore, we employ ground truth information to remove incorrect feature matches before using them to compute the robot's pose. Consequently, we are able to set the maximum number of outliers to expect to 0.

Correct (%)	Volume (m^3)	Area (m^2)	Rotation accuracy (°)	Features w. depth	Distance per keyframe (m)	Inconsistencies (%)
100	0.83	0.57	0.83	90	14.13	33.5

Table 8.32: Quantitative results of the interval-based visual-LiDAR odometry approach assuming no image features are mismatched.

The results are depicted in Table 8.32. As expected, we achieve more accurate results and have to insert fewer keyframes because we can trust the information of each image feature match, rather than conservatively assuming that some of them might be outliers. Naturally, the number of features with depth information decreases since fewer keyframes are inserted and outliers are removed beforehand. Still, the solution is correctly enclosed for all 2644 image frames.

In addition, more inconsistencies can be detected due to the more accurate results. Again, we want to point out that we provide the same image features for both our interval-based and the established stochastic approach. Consequently, in this experiment the outliers are removed also for the stochastic approach. Nevertheless, it is not able to compute a reasonable pose estimate for every third image frame. Thus, we are once again able to show the importance of error modeling during sensor fusion for robust visual-LiDAR odometry.

In the future, more reliable feature matching algorithms should be investigated. Here, the main focus should be on the prevention of outliers instead of increasing the number of image feature matches. As our experiments show, a few reliable image feature matches (sometimes as few as ten) for which we can find accurate depth information suffice to compute a reasonable accurate pose box. This is in contrast to the stochastic approach which generally requires more image feature matches since it cannot assess the accuracy of the depth estimates. If only a few image feature matches are present and the depth estimate of one of those features is considerably wrong, the optimization process is significantly distorted. Consequently, we also believe that our approach prevails in situations in which only a few image feature matches can be found.

9 Summarizing Discussion and Prospects

The experimental evaluation aimed to determine whether the newly introduced approaches can increase the integrity of autonomous vehicles by providing guarantees and giving timely warnings to the user if the information of the system is too uncertain to ensure safe operation. Moreover, a comparison with established stochastic approaches was provided since we introduce our approaches in the context of interval-based error modeling, which constitutes a fundamentally different approach to take sensor errors into account.

In general, we were able to show that our approaches for the spatiotemporal calibration between camera, laser scanner and IMU are suitable to compute intervals that, on the one hand, enclose the true calibration parameters and, on the other hand, allow to assess the accuracy of the calibration. Moreover, we demonstrated the importance of correctly modeling sensor and inter-sensor errors for the subsequent sensor fusion. By considering those uncertainties, our new visual-LiDAR odometry approach is able to compute 6 DOF pose estimates that are guaranteed to contain the true solution. Consequently, it can identify faults of established stochastic approaches that do not consider these uncertainties.

Spatiotemporal calibration between camera and IMU

In Section 8.1 we evaluated our interval-based approach for finding the extrinsic rotation between camera and IMU as well as the constant time offset between the clocks of the sensors. Our experiments show that we are able to compute a reasonably accurate enclosure for the time offset that can be employed to fuse information from the camera and IMU. However, the extrinsic rotation is not accurate enough to allow sensor fusion, and thus we present prospects to further improve the results.

First, we employed simulated data to compare our approach to ground truth information, which cannot be obtained for real data, and to evaluate the results for different time offsets. The evaluation of our newly introduced time offset contractor $\mathcal{C}_{\text{offset}}$ shows that we are able to enclose the true time offset in a guaranteed way, meaning that the true offset always resides in the computed intervals. Naturally, the width of these intervals and thus the accuracy of the time offset depends on the accuracy of the estimate for the extrinsic rotation as well as the velocity and the axis of the rotation of our multi-sensor system. However, it is independent of the true time offset. The evaluation of the full spatiotemporal calibration approach shows that we are able to reliably enclose not only the time offset but also the extrinsic rotation.

The evaluation using real data confirms the results of our simulation studies, thus showing the applicability of our approach to real multi-sensor systems. The achieved accuracies - $w([\tau]) = 8.8\,\text{ms}$ for the time offset and $w([\theta]) = 12.2°$ for the extrinsic calibration - are similar. A comparison with an established stochastic approach shows that our results are plausible since the results of the stochastic approach are enclosed in our intervals. However, the established approach does not propagate the sensor errors to the final calibration result. Therefore, the accuracy of the result can only be inferred from previous experiments in which the true solution is compared to the computed result. Here, a large number of experiments are required to determine a stochastic distribution, which consequently only applies to the specific sensor combination and the experimental setup. Although the accuracy of the results of our approach is significantly lower due to the consideration of the worst case, we are, to the best of our knowledge, for the first time able to guarantee the calibration results and to infer the accuracy directly from the sensor errors.

As the evaluation of our approach for bounded-error visual-LiDAR odometry shows, the accuracy of the time offset is adequate to enable information fusion from camera and IMU. However, the extrinsic rotation between both sensors is too inaccurate to allow a useful transformation of data measured by the IMU into the coordinate system of the camera. Consequently, in the future, our approach should be improved with regard to the accuracy of the extrinsic rotation. A first improvement could be to employ tubes enclosing the angular velocities of both sensors instead of tubes enclosing the orientation of the sensors. This would allow a longer experiment duration and thus enable to gather more data. At the moment, the duration of the experiments is limited by the rapidly increasing uncertainty of the orientation estimates caused by the integration of the angular velocities of the IMU. Consequently, after a few seconds, the intervals are too wide to constrain the spatiotemporal calibration parameters. However, tubes that are guaranteed to enclose the angular velocities of the camera cannot be computed straightforwardly due to the low frequency of measurements.

Moreover, our experiments show that we are not able to eliminate the dependencies between the extrinsic rotation parameters. We believe the reason is that we were not able to rotate the multi-sensor system freely due to the requirement to keep the calibration target in the field of view of the camera. Consequently, either multiple calibration targets should be employed in the future, or the approach should be adapted to not rely on a calibration target.

Extrinsic calibration between camera and LiDAR

Section 8.2 provided an evaluation of our interval-based approach that computes intervals enclosing the extrinsic transformation parameters between camera and laser scanner. Our experiments using both simulated and real data show the applicability of our approach to the extrinsic calibration problem and highlight the advantages over an established stochastic method.

First, an experiment using simulated data shows the influence of the checkerboard pose on the accuracy of the 6 DOF calibration. This experiment highlights a first advantage over the established stochastic approach. While our method allows assessing the accuracy of each transformation parameter, and thus enables the user to determine whether additional checkerboard poses are required, the stochastic approach does not compute the accuracy of the

parameters. Consequently, the user must acquire data for a sufficient number of checkerboard poses, but cannot track the accuracy, and therefore cannot be sure that the calibration is sufficiently accurate.

Next, our results show that six checkerboard poses suffice to compute a reasonably accurate enclosure of the parameters. This is a textbook example of how the different contractors for individual checkerboard poses can be combined to produce an even more powerful contractor. Adding even more checkerboard poses results in only a slight improvement of the results. This again highlights the need to assess the parameter accuracies during the calibration to determine whether a checkerboard pose adds valuable information. The final accuracy of our calibration experiment is 0.7° for the rotation around the x- and y-axis, 0.4° for the rotation around the z-axis, approximately 3.7 cm along the x- and y-axis and 2.1 cm along the z-axis. As expected, the parameter domains contain the true transformation parameters. This is true for all experiments using simulated data, making us confident that our approach is guaranteed to enclose the true result.

Naturally, these accuracies seem disappointing when comparing them to the accuracies reported by state-of-the-art approaches. It should be noted, however, that the results of interval-based approaches reflect the worst case, and therefore cannot be directly compared to stochastic approaches. Moreover, the evaluation of our bounded-error visual-LiDAR odometry shows that an even lower accuracy is sufficient to perform reasonably accurate dead reckoning. Nevertheless, improving the accuracy of the extrinsic calibration would allow to also fuse information from camera and laser scanner more accurately. Possible improvements of our approach could be made by incorporating more constraints, e.g. by using different calibration targets.

In addition, we conducted a parameter study in which we varied the sensor errors and the number of outliers. The results show that the uncertainty of our result increases with an increasing sensor error. However, it only increases slightly, which is insignificant to the extent to which we increased the sensor errors. Nevertheless, that only applies if the exact error bounds are known and the actual error is not overestimated. This is further illustrated by our experiment in which we randomly simulated different numbers of outliers, but only assumed the maximum number of outliers to be known. Here, the uncertainty increases significantly. If, on the other hand, exactly as many outliers are simulated as are assumed for the relaxed intersection, the uncertainty does not increase. This again shows the importance of knowing the exact error bounds (cf. Section 3.11), which constitutes a major weakness of interval-based approaches.

To highlight another advantage of our interval-based approach over stochastic approaches, we also conducted an experiment in which we simulated a systematic error for the distance measurements of the laser scanner. Since a systematic error is not consistent with the assumption of zero-mean errors, the result of the stochastic approach deviates significantly from the true result. In contrast, our interval-based approach is not affected and computes the same result regardless of whether there is a systematic error or not. This constitutes a major advantage of our and interval-based approaches in general since systematic errors often occur for the distance measurements of a laser scanner due to, for example, an imprecise intrinsic calibration (cf. Section 4.1.1).

Moreover, we are also able to show the applicability of our approach to a real multi-sensor system. Here, the resulting accuracies of the extrinsic calibration parameters are similar but slightly lower than in our simulation studies. This can be attributed to less optimal sensor error bounds, which we only derived from the manufacturer's information in the sensor data sheets and which probably overestimate the actual error. As explained previously, determining optimal error bounds is a major challenge of interval-based approaches. Nevertheless, the results are still guaranteed. Moreover, the parameters computed using the stochastic approach do not reside in the corresponding intervals, and are therefore presumably incorrect. This could be due to systematic sensor errors and shows the ability of our approach to detect inconsistencies.

Finally, in the future, our approach could be extended to not rely on a calibration target, so it can be employed to automatically re-calibrate the multi-sensor system during operation.

Bounded-error visual-LiDAR odometry

In Section 8.3 we evaluated our approach that fuses information from camera, laser scanner and IMU to perform dead reckoning, i.e. compute the robot's 6 DOF pose incrementally in a locally defined coordinate system. Our experiments show that we are able to compute a box enclosing the true pose for all image frames, and thus the desired guarantees are achieved.

Naturally, the results of our approach for bounded-error visual-LiDAR odometry are rather conservative (i.e. less accurate) compared to established stochastic approaches. However, this is expected since the results are guaranteed. Moreover, the uncertainty of our pose estimates increases rapidly due to the nature of dead-reckoning approaches, since no technique to eventually reduce the accumulated uncertainty is used. Consequently, whenever a new keyframe is inserted, the previous uncertainty is propagated to the new pose estimates. For our experiments, already after 10 s the box containing the position of our vehicle covers an area of $18\,\mathrm{m}^2$ and has a volume of $52\,\mathrm{m}^3$. However, this is expected since we only perform visual-LiDAR odometry. Nevertheless, in the future, our approach should be augmented by global contractors that keep the uncertainty from growing infinitely. These global contractors can be found, for example, by extending our approach by bundle adjustment, extending it to SLAM, or by using GNSS information.

A general weakness of interval-based approaches is the dependence on correct error bounds. As explained in Section 3.11, the bounds should be guaranteed to enclose the true result, but should not be too large to avoid overestimating the actual error. Our parameter studies, which show the results for different sensor error bounds or maximum outlier percentages, repeatedly highlight this phenomenon. Thus, future work should focus on the extension of our introduced bounded error sensor models to even more accurately model possible error sources. Moreover, more extensive information on the maximum sensor errors is required from the sensor manufacturers. Besides, although the accuracies achieved using our spatiotemporal calibration approaches are generally sufficient for reasonably accurate sensor fusion, they can still be improved as detailed in the previous paragraphs. This, in turn, would also increase the accuracy of our visual-LiDAR odometry approach.

Furthermore, our experiments show that we are able to compute more accurate pose estimates if feature matching outliers are removed beforehand and a relaxed intersection is not required. In fact, in most image frames there are no wrongly matched image features. This

can be seen by our experiment in which we do not remove outliers beforehand but nevertheless set the maximum number of outliers to 0. Here, the result is still correct for 96.86 % of all image frames. However, since we do not know for which image frames the outliers occur, we have to assume outliers for every image frame, and thus overestimate the actual error for most image frames. Consequently, guaranteed feature matching algorithms should be developed that aim to not generate outliers, rather than maximizing the number of image feature matches.

An advantage of our approach is the possibility to dynamically insert keyframes whenever the uncertainty of the pose estimate grows too large. This is made possible by the fact that our approach not only computes a solution but a set of solutions, which at the same time reflects the current uncertainty. Consequently, the insertion of keyframes is no longer heuristic but based on the actual uncertainty of the system. The experiments show that the pose estimation in areas in which fewer features can be assigned precise depth information (e.g. due to the discontinuity of the environment) is, as expected, more uncertain. Therefore, the presented approach for visual-LiDAR odometry is particularly suitable for safety-critical systems such as autonomous vehicles, for which the worst-case error is more important than the average error and which cannot be based on a heuristic keyframe insertion strategy.

However, an obvious disadvantage due to the design of interval-based approaches is the inability to compute the most likely point-valued solution. We only compute a set of solutions which is guaranteed to contain the true solution, but is it not possible find this true solution. Nevertheless, our experiments show that the midpoint of the intervals can serve as a reasonably accurate solution. Still, a combination of interval-based and stochastic approaches will be required in the future to combine the advantages of both worlds.

Finally, we would like to emphasize that we are able to meet our initial goal of increasing the integrity of autonomous vehicles by providing warnings that notify the user if the result of an established stochastic approach is incorrect, or the uncertainty has become too large. For our dataset we are able to identify an inconsistency of the stochastic approach in approximately 30 % of all image frames although the same image features, point clouds and spatiotemporal calibration parameters are provided to both the stochastic and our interval-based approach. This shows the importance of correctly modeling all errors during sensor fusion in order to assess the accuracy of the 3D features when employing them to compute the motion estimation. This is particularly important if only a few image features are available and accurate depth information can be assigned to even fewer. In these cases the most inconsistencies of the established approach are detected. In contrast, our interval-based approach manages to compute a box containing the true solution because it can assess the accuracy of the 3D features and thus weight them. Naturally, further methods could be employed to hopefully correct the errors made during the optimization of the stochastic approach (e.g. bundle adjustment). However, we believe it is advantageous not to make these errors in the first place.

10 Conclusions

This thesis considers the question of how robust dead reckoning, i.e. incremental localization without a map of the environment or GNSS information, of mobile robots in 3D can be carried out. Especially for autonomous driving, the question arises of how, in addition to being robust, the safety and integrity of each autonomous vehicle can be guaranteed. Fig. 10.1 depicts the overview that was established throughout the thesis.

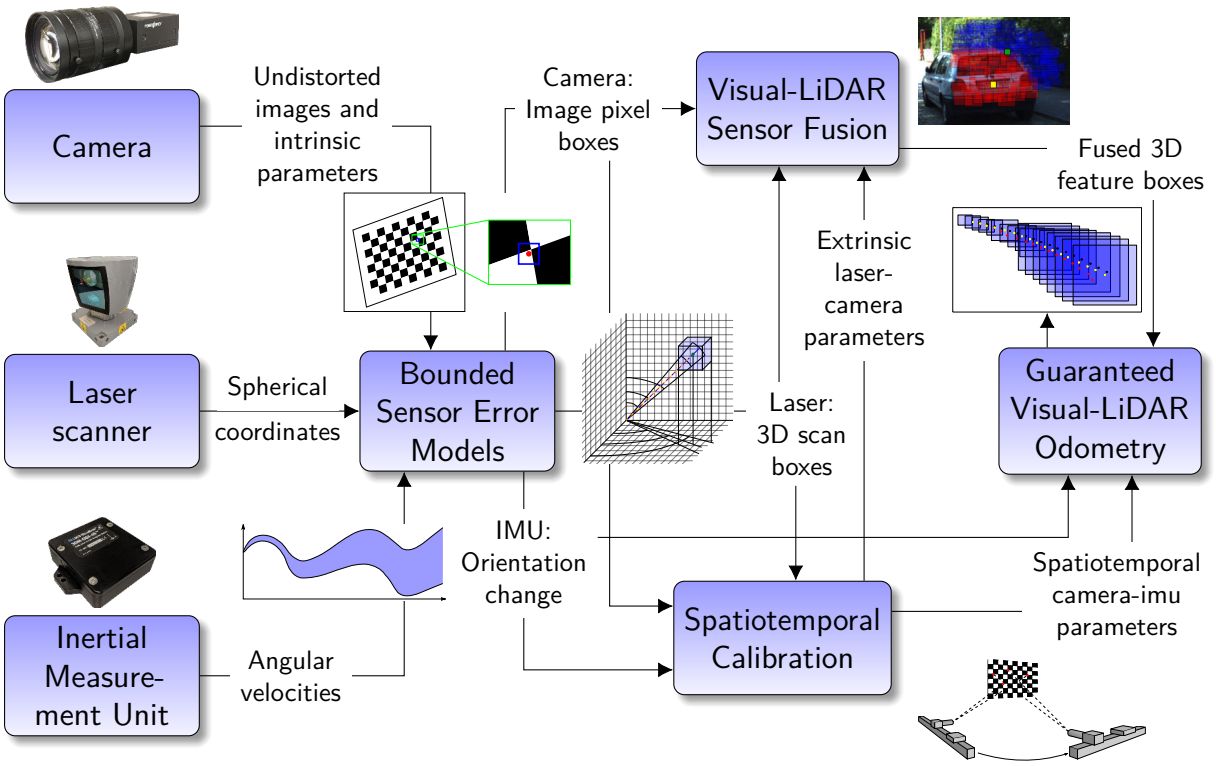


Figure 10.1: Overview of the contributions of this thesis.

To increase the robustness with respect to sensor failures and improve the trust in the information (i.e. integrity) of autonomous systems, it is beneficial to fuse information from multiple sensors. The present work therefore deals with the fusion of information from three sensors commonly used in mobile robotics, namely the camera, the laser scanner and the IMU. Besides the sensor fusion, this thesis also tackles the question of how results can be guaranteed under the consideration of all possible error sources and how simultaneously the uncertainties of these results can be assessed. Here, the ultimate goal is to provide timely warnings to the user if the uncertainty is too large, or to identify inconsistencies of an established approach.

Therefore, this thesis first presents newly developed bounded error models for each of the three sensors. These error models take into account all identified types of error and bound them using intervals. The main advantages of interval analysis are the inherent guarantees that are required for safety-critical systems and the usually unknown true error distribution of sensors, which means that the assumption of any stochastic distribution is generally incorrect without a proper calibration of the sensor. Systematic errors in particular cannot be reconciled with the zero-mean errors assumption, which is essential for all stochastic approaches.

For robust sensor fusion, however, it is not sufficient to only consider the different errors of each sensor, since also the inter-sensor properties required for sensor fusion (e.g. a time offset between sensor clocks and the extrinsic transformation between sensor coordinate systems) cannot be perfectly determined and therefore induce additional uncertainties. In general, and in particular in the context of interval analysis, however, there are no approaches to perform a spatiotemporal calibration between camera, laser scanner and IMU, which take into account the sensor errors and propagate them to the resulting calibration parameters. Consequently, interval-based approaches for the spatiotemporal calibration between camera, laser scanner and IMU are presented in this work.

For the spatiotemporal calibration between camera and IMU, the main contribution of this thesis is a newly developed time offset contractor ($\mathcal{C}_{\text{offset}}$) that can be employed to efficiently compute a constant time offset between two tubes. Although this work shows the application of the time offset contractor ($\mathcal{C}_{\text{offset}}$) to the problem of sensor calibration, it is a novel general contractor in the context of constraint programming over dynamical systems that can be applied to any two tubes that are delayed in time. Furthermore, using the new error models, this work transfers existing methods into the interval context to determine tubes for the camera and IMU, which describe the same physical phenomenon, i.e. the rotation of both sensors. This includes the transfer of the PnP problem into a CSP for which Gauss-Seidel and forward-backward contractors are built. The corresponding experiments using both simulated and real data show the ability of the time offset contractor to robustly enclose the true time offset. The accuracy is sufficient for the following sensor fusion. While the extension to also compute the extrinsic rotation between the two sensors can also reliably enclose the true rotation, the computed intervals are too inaccurate to be used for sensor fusion. The reason is that due to the integration of angular velocities, the uncertainty of the system increases rapidly. Therefore, an approach should be developed in the future that avoids this integration.

For the extrinsic calibration between camera and LiDAR, this thesis contributes interval-based methods to extract checkerboard features from both camera and laser scan data. These checkerboard features constrain the transformation between the sensors. Subsequently, forward-backward contractors are built that enable the transformation to be computed without initial values. As the corresponding experiments show, the approach is capable of reliably enclosing the true 6-DOF transformation parameters for both simulated and real data. This is valid regardless of the configuration of the multi-sensor system and the actual sensor errors, provided that the correct error bounds are known. Moreover, the results are sufficiently accurate so that they can subsequently be used for sensor fusion.

Together with the bounded sensor error models, these two approaches for the spatiotemporal calibration of the three sensors enable all possible error sources for sensor fusion to be determined.

On this basis, an approach to interval-based visual-LiDAR odometry is developed to answer the initial question of guaranteed and robust dead reckoning. Besides the errors of each individual sensor, an important contribution of this thesis is the consideration of the uncertainties of the spatiotemporal calibration parameters. Even if methods other than the approaches presented for spatiotemporal calibration are used, it is advisable to assume unknown but bounded errors for the calibration parameters. The reason is that the error of these parameters does not follow a stochastic distribution due to the rigidity of the system - since the multi-sensor system is mounted rigidly and the time offset is assumed to be constant - and thus any deviations of the computed parameters from the true parameters lead to a purely systematic error during sensor fusion. Naturally, the calibration cannot be perfect, and therefore a systematic error, which is not compatible with stochastic methods, inevitably occurs.

The developed approach for interval-based visual-LiDAR odometry fuses information from camera and laser scanner in the sense that image features are assigned depth information from the laser scanner. The novelty of the present work is to not only compute the depth of image features, but to also determine the uncertainty of these depth estimates. This enables to assess the trustworthiness of 3D features. Trustworthy 3D features constrain the subsequent pose estimation stronger than untrustworthy features, and thus contribute more to the final pose computation. In addition, the presented approach succeeds in finding an initial estimate of the rotation of the vehicle that is guaranteed to enclose the true rotation. For this purpose, the angular velocities measured by the IMU are integrated, taking into account all sensor errors. Afterwards, the resulting rotation intervals are transformed into the time and coordinate system reference of the camera, taking into account all uncertainties of the spatiotemporal calibration parameters.

The evaluation using real data shows that the introduced approach is capable of providing guaranteed boxes enclosing the true 6-DOF pose in $100\,\%$ of all cases. Moreover, the uncertainty of each pose estimate reflects the quality of the corresponding 3D features. Besides, the experiments show the applicability of the approach to identify inconsistencies of an established method. Even if only a few accurate 3D features are available, the new approach manages to compute a sufficiently accurate and in particular correct pose estimate. In contrast, the comparable state-of-the-art method fails in these situations because it is unable to assess the accuracies of the depth estimates and use them to subsequently weight the features during optimization. Thus, the initial requirement to provide the user with timely warnings in the event of insufficiently accurate localization results or errors in the stochastic approach is met. Nonetheless, during a parameter study possible improvements of the approach are determined. In particular, reducing the number of outliers during image feature matching and determining the sensor and inter-sensor error bounds more accurately prove to be promising.

In summary, this thesis shows that a guaranteed and robust sensor fusion is possible and particularly useful for safety-critical systems. However, the accuracy of the spatiotemporal calibration and the pose estimates can still be improved, with the determination of sensor error bounds being a particular problem. Therefore, future research should deal with the experimental determination or derivation of these sensor error bounds from physical effects. Moreover, interval-based approaches are valuable to avoid computing a completely wrong solution. In the future, however, a combination with stochastic approaches will be required to select the most likely point-valued solution in the interval.

Bibliography

[1] Kraftfahrt-Bundesamt, "Verkehr in Kilometern – Inländerfahrleistung 2018," https://www.kba.de/DE/Statistik/Kraftverkehr/VerkehrKilometer/pseudo_verkehr_in_kilometern_node.html, Last accessed on 02/17/20.

[2] M. Wörner, F. Schuster, F. Dolitzscher, C. G. Keller, M. Haueis, and K. Dietmayer, "Integrity for autonomous driving: A survey," in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, Savannah, Georgia, USA, Apr. 2016.

[3] ISO 26262-1:2011, "Road vehicles – Functional safety – Part 1: Vocabulary," International Organization for Standardization, Tech. Rep., 2011.

[4] International Civil Aviation Organization, "Aeronautical Telecommunications," in *Annex 10 to the Convention on International Civil Aviation*, Montreal, Quebec, Canada, 2006.

[5] H. Strasdat, J. M. M. Montiel, and A. Davison, "Scale Drift-Aware Large Scale Monocular SLAM," in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, Jun. 2010.

[6] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, Eds., *Autonomes Fahren*. Springer Nature, 2015.

[7] G. Meyer and S. Beiker, Eds., *Road Vehicle Automation 3*. Springer International Publishing, 2016.

[8] R. P. Feynman and F. Dyson, *The Pleasure of Finding Things Out: The Best Short Works of Richard P. Feynman*, J. Robbins, Ed. Helix Books, 2005.

[9] L. Jaulin, M. Kieffer, O. Didrit, and É. Walter, *Applied Interval Analysis*. Springer London, 2001.

[10] V. Kreinovich, "Data Processing Beyond Traditional Statistics: Applications of Interval Computations. A Brief Introduction," *A Supplement to the international journal of Reliable Computing*, 1995.

[11] M. Grabe, *Measurement Uncertainties in Science and Technology*. Springer International Publishing, 2014.

[12] H. Kutterer and S. Schön, "Alternativen bei der Modellierung der Unsicherheit beim Messen," in *Geodätische Woche*, Stuttgart, Germany, Oct. 2004.

[13] S. Schön, "Analyse und Optimierung geodätischer Messanordnungen unter besonderer Berücksichtigung des Intervallansatzes," PhD thesis, Universität Fridericiana zu Karlsruhe, 2003.

[14] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," *Lecture Notes in Mathematics*, pp. 105–116, 1978.

[15] D. Mills, "Internet time synchronization: The network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[16] C. Glennie, "Calibration and Kinematic Analysis of the Velodyne HDL-64E S2 Lidar Sensor," *Photogrammetric Engineering & Remote Sensing*, vol. 78, no. 4, pp. 339–347, Apr. 2012.

[17] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[18] G. Healey and R. Kondepudy, "Radiometric CCD camera calibration and noise estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 267–276, 1994.

[19] P. D. Groves, *Principles of GNSS, Inertial, and Multi-Sensor Integrated Navigation Systems (GNSS Technology and Applications)*. Artech House Publishers, 2007.

[20] J. A. Farrell, *Aided Navigation: GPS with High Rate Sensors*. McGraw-Hill Education, 2008.

[21] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, Feb. 2017.

[22] A. Noureldin, T. B. Karamat, and J. Georgy, *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration*. Springer-Verlag Berlin Heidelberg, 2012.

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[24] H. Ku, "Notes on the use of propagation of error formulas," *Journal of Research of the National Bureau of Standards, Section C: Engineering and Instrumentation*, vol. 70C, no. 4, p. 263, Oct. 1966.

[25] T. F. Wiener, "Theoretical analysis of gimballess inertial reference equipment using delta-modulated instruments," PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1962.

[26] M. D. Shuster, "A Survey of Attitude Representations," *Journal of the Astronautical Sciences*, vol. 41, no. 4, pp. 439–517, Oct. 1993.

[27] R. Voges and B. Wagner, "Timestamp Offset Calibration for an IMU-Camera System Under Interval Uncertainty," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

[28] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Pr., 2003.

[29] J. C. McGlone, *Manual of Photogrammetry - Sixth Edition*. ASPRS, 2013.

[30] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, Aug. 2003.

[31] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, Jul. 2008.

[32] L. Kneip, H. Li, and Y. Seo, "UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability," in *European Conference on Computer Vision (ECCV)*, Zurich, Switzerland, Sep. 2014, pp. 127–142.

[33] Y. I. Abdel-Aziz and H. M. Karara, "Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry," *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 2, pp. 103–107, Feb. 2015.

[34] S. Li, C. Xu, and M. Xie, "A Robust O(n) Solution to the Perspective-n-Point Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1444–1450, Jul. 2012.

[35] E. Olson, "A passive solution to the sensor synchronization problem," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, Oct. 2010.

[36] M. Li and A. I. Mourikis, "Online temporal calibration for camera–IMU systems: Theory and algorithms," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 947–964, 2014.

[37] T. Qin and S. Shen, "Online Temporal Calibration for Monocular Visual-Inertial Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

[38] J. Kelly, N. Roy, and G. S. Sukhatme, "Determining the Time Delay Between Inertial and Visual Sensor Measurements," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1514–1523, 2014.

[39] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013.

[40] J. Rehder and R. Siegwart, "Camera/IMU calibration revisited," *IEEE Sensors Journal*, vol. 17, no. 11, pp. 3257–3268, 2017.

[41] J. Rehder, R. Siegwart, and P. Furgale, "A general approach to spatiotemporal calibration in multisensor systems," *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 383–398, 2016.

[42] R. Voges, C. S. Wieghardt, and B. Wagner, "Finding Timestamp Offsets for a Multi-Sensor System Using Sensor Observations," *Photogrammetric Engineering & Remote Sensing*, vol. 84, no. 6, pp. 357–366, 2018.

[43] J. Levinson and S. Thrun, "Automatic Online Calibration of Cameras and Lasers," in *Robotics: Science and Systems (RSS)*, Berlin, Germany, Jun. 2013.

[44] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, Oct. 2007.

[45] J. Kümmerle, T. Kühner, and M. Lauer, "Automatic Calibration of Multiple Cameras and Depth Sensors with a Spherical Target," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

[46] Y. Park, S. Yun, C. Won, K. Cho, K. Um, and S. Sim, "Calibration between Color Camera and 3D LIDAR Instruments with a Polygonal Planar Board," *Sensors*, vol. 14, no. 3, pp. 5333–5353, Mar. 2014.

[47] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan: IEEE, Sep. 2004.

[48] R. Unnikrishnan and M. Hebert, "Fast Extrinsic Calibration of a Laser Rangefinder to a Camera," Tech. Rep., 2005.

[49] L. Zhou, Z. Li, and M. Kaess, "Automatic Extrinsic Calibration of a Camera and a 3D LiDAR Using Line and Plane Correspondences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

[50] L. Zhou and Z. Deng, "Extrinsic calibration of a camera and a lidar based on decoupling the rotation from the translation," in *IEEE Intelligent Vehicles Symposium*, Alcala de Henares, Spain, Jun. 2012.

[51] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: A survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 55–81, Nov. 2012.

[52] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality*, Washington, DC, USA, Nov. 2007.

[53] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[54] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.

[55] I. Cvišić, J. Ćesić, I. Marković, and I. Petrović, "SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles," *Journal of Field Robotics*, vol. 35, no. 4, pp. 578–595, Nov. 2017.

[56] M. Buczko and V. Willert, "Flow-Decoupled Normalized Reprojection Error for Visual Odometry," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, Nov. 2016.

[57] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, Dec. 2014.

[58] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," *Robotics Research. Springer Tracts in Advanced Robotics*, vol. 100, pp. 235–252, Aug. 2016.

[59] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *IEEE International Conference on Advanced Robotics (ICAR)*, Istanbul, Turkey, Jul. 2015.

[60] J. Zhang, M. Kaess, and S. Singh, "Real-time Depth Enhanced Monocular Odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, Illinois, USA, Sep. 2014.

[61] J. Zhang, M. Kaess, and S. Singh, "A real-time method for depth enhanced visual odometry," *Auton Robot*, Dec. 2015.

[62] J. Zhang and S. Singh, "Visual-lidar Odometry and Mapping: Low-drift, Robust, and Fast," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, May 2015.

[63] J. Zhang and S. Singh, "Enabling aggressive motion estimation at low-drift and accurate mapping in real-time," in *IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, Singapore, May 2017.

[64] J. Zhang and S. Singh, "Laser-visual-inertial odometry and mapping with high robustness and low drift," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, Aug. 2018.

[65] J. Graeter, A. Wilczynski, and M. Lauer, "LIMO: Lidar-Monocular Visual Odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

[66] R. E. Moore, *Interval Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1966.

[67] T. L. Heath, *The Works of Archimedes*. Dover, New York, 1897.

[68] S. Rohou, "Reliable robot localization: A constraint-programming approach over dynamical systems," PhD thesis, ENSTA Bretagne/Lab-STICC (France) and The University of Sheffield (England), 2017.

[69] G. Chabert, "IBEX, a C++ library for constraint processing over real numbers," http://www.ibex-lib.org, 2017.

[70] V. Kreinovich, "Interval Computations and Interval-Related Statistical Techniques: Tools for Estimating Uncertainty of the Results of Data Processing and Indirect Measurements," in *Data Modeling for Metrology and Testing in Measurement Science*, Birkhäuser Boston, Nov. 2008, pp. 1–29.

[71] G. W. Walster and V. Kreinovich, "For unknown-but-bounded errors, interval estimates are often better than averaging," *ACM SIGNUM Newsletter*, vol. 31, no. 2, pp. 6–19, Apr. 1996.

[72] G. Chabert and L. Jaulin, "Contractor programming," *Artificial Intelligence*, vol. 173, no. 11, pp. 1079–1100, Jul. 2009.

[73] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget, "Revising Hull and Box Consistency," in *16th International Conference on Logic Programming*, Dakar, Senegal, Apr. 1999.

[74] L. Jaulin and E. Walter, "Set inversion via interval analysis for nonlinear bounded-error estimation," *Automatica*, vol. 29, no. 4, pp. 1053–1064, Jul. 1993.

[75] L. Jaulin, É. Walter, and O. Didrit, "Guaranteed Robust Nonlinear Parameter Bounding," in *IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, Lille, France, Jul. 1996.

[76] L. Jaulin and E. Walter, "Guaranteed robust nonlinear minimax estimation," *IEEE Transactions on Automatic Control*, vol. 47, no. 11, pp. 1857–1864, Nov. 2002.

[77] A. Bethencourt and L. Jaulin, "Solving non-linear constraint satisfaction problems involving time-dependant functions," *Mathematics in Computer Science*, vol. 8, no. 3-4, pp. 503–523, 2014.

[78] S. Rohou, "Tubex: a C++ library providing tools for computations over sets of trajectories," http://www.simon-rohou.fr/research/tubex-lib, 2018.

[79] K. Marzullo and S. Owicki, "Maintaining the time in a distributed system," *ACM SIGOPS Operating Systems Review*, vol. 19, no. 3, pp. 44–54, 1985.

[80] O. Bezet and V. Cherfaoui, "On-Line Timestamping Synchronization in Distributed Sensor Architectures," in *IEEE Real Time and Embedded Technology and Applications Symposium*, San Francisco, CA, USA, Mar. 2005.

[81] M. Zaman and J. Illingworth, "Interval-Based Time Synchronisation of Sensor Data in a Mobile Robot," in *Intelligent Sensors, Sensor Networks and Information Processing Conference*, Melbourne, Vic., Australia, Australia, Dec. 2004.

[82] A. J. Terry, M. Zaman, and J. Illingworth, "Sensor fusion by a novel algorithm for time delay estimation," *Digital Signal Processing*, vol. 22, no. 3, pp. 439–452, May 2012.

[83] M. Langerwisch and B. Wagner, "Guaranteed Mobile Robot Tracking Using Robust Interval Constraint Propagation," in *International Conference on Intelligent Robotics and Applications (ICIRA)*, Montréal, Québec, Canada, Oct. 2012.

[84] I.-F. Kenmogne, V. Drevelle, and E. Marchand, "Image-based UAV localization using Interval Methods," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, Sep. 2017.

[85] I.-F. Kenmogne, V. Drevelle, and E. Marchand, "Interval-Based Cooperative Uavs Pose Domain Characterization from Images and Ranges," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

[86] 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.

[87] L. Jaulin, "Range-only SLAM with indistinguishable landmarks; a constraint programming approach," *Constraints*, vol. 21, no. 4, pp. 557–576, Oct. 2015.

[88] B. Vincke, A. Lambert, and A. Elouardi, "Guaranteed simultaneous localization and mapping algorithm using interval analysis," in *13th International Conference on Control Automation Robotics & Vision (ICARCV)*, Singapore, Singapore, Dec. 2014.

[89] 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, Feb. 2013.

[90] M. Mustafa, A. Stancu, S. P. Guteirrez, E. A. Codres, and L. Jaulin, "Rigid Transformation Using Interval Analysis for Robot Motion Estimation," *International Conference on Control Systems and Computer Science (CSCS)*, May 2015.

[91] M. Mustafa, A. Stancu, N. Delanoue, and E. Codres, "Guaranteed SLAM – An interval approach," *Robotics and Autonomous Systems*, vol. 100, pp. 160–170, Feb. 2018.

[92] P. Skrzypczynski, "How to Recognize and Remove Qualitative Errors in Time-of-Flight Laser Range Measurements," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, Sep. 2008.

[93] Q. Wang, N. Zissler, and R. Holden, "Evaluate error sources and uncertainty in large scale measurement systems," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 1, pp. 1–11, Feb. 2013.

[94] C. Glennie and D. D. Lichti, "Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning," *Remote Sensing*, vol. 2, no. 6, pp. 1610–1624, Jun. 2010.

[95] C. Glennie, "Rigorous 3D error analysis of kinematic scanning LiDAR systems," *Journal of Applied Geodesy*, vol. 1, no. 3, pp. 147–157, 2007.

[96] Q. Pentek, T. Allouis, O. Strauss, and C. Fiorio, "Developing And Validating a Predictive Model of Measurement Uncertainty For Multi-beam LiDARs: Application to the Velodyne VLP-16," in *International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Xi'an, China, Nov. 2018.

[97] M. Langerwisch, "Kartierung und Lokalisation durch mobile Serviceroboter unter der Annahme unbekannter aber begrenzter Sensorfehler," PhD thesis, Leibniz Universität Hannover, 2014.

[98] B. Telle, M.-J. Aldon, and N. Ramdani, "Camera calibration and 3D reconstruction using interval analysis," in *International Conference on Image Analysis and Processing (ICIAP)*, Mantova, Italy: Mantova, Italy, Sep. 2003.

[99] B. Kamgar-Parsi and B. Kamgar-Parsi, "Evaluation of quantization error in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 9, pp. 929–940, Sep. 1989.

[100] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello, "A Visual Odometry Framework Robust to Motion Blur," in *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009.

[101] B. T. Koik and H. Ibrahim, "A Literature Survey on Blur Detection Algorithms for Digital Imaging," in *1st International Conference on Artificial Intelligence, Modelling and Simulation*, Kota Kinabalu, Malaysia, Dec. 2013.

[102] C. Liu, W. T. Freeman, R. Szeliski, and S. B. Kang, "Noise Estimation from a Single Image," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, NY, USA, Jun. 2006.

[103] Z. Xing and D. Gebre-Egziabher, "Modeling and bounding low cost inertial sensor errors," in *IEEE/ION Position, Location and Navigation Symposium*, Monterey, CA, USA, May 2008.

[104] D. Unsal and K. Demirbas, "Estimation of deterministic and stochastic IMU error parameters," in *IEEE/ION Position, Location and Navigation Symposium*, Myrtle Beach, SC, USA, Apr. 2012.

[105] D. Tedaldi, A. Pretto, and E. Menegatti, "A robust and easy to implement method for IMU calibration without external equipments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[106] W. S. Flenniken, J. H. Wall, and D. M. Bevly, "Characterization of Inertial Sensor Measurements for Navigation Performance Analysis," in *19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS)*, Fort Worth, TX, USA, Sep. 2006.

[107] S. K. Das, D. Pal, V. Kumar, S. Nandy, K. Banerjee, and C. Mazumdar, "Stochastic Characterization of a MEMs based Inertial Navigation Sensor using Interval Methods," *International Journal of Image, Graphics and Signal Processing*, vol. 7, no. 7, pp. 24–32, Jun. 2015.

[108] B. Telle, M.-J. Aldon, and N. Ramdani, "Guaranteed 3D visual sensing based on interval analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, Oct. 2003.

[109] B. Telle, O. Stasse, K. Yokoi, T. Ueshiba, and F. Tomita, "Three Characterizations of 3D Reconstruction Uncertainty with Bounded Error," in *IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, Apr. 2005.

[110] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, "Solving Interval Linear Systems is NP-Hard," in *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Springer US, 1998, pp. 99–110.

[111] L. Jaulin and E. Walter, "Guaranteed parameter bounding for nonlinear models with uncertain experimental factors," *Automatica*, vol. 35, no. 5, pp. 849–856, May 1999.

[112] R. Voges and B. Wagner, "Extrinsic Calibration Between a 3D Laser Scanner and a Camera Under Interval Uncertainty," in *Book of Abstracts of the 12th Summer Workshop on Interval Methods (SWIM 2019)*, Palaiseau, France, Jul. 2019.

[113] 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)*, Las Vegas, NV, USA, Oct. 2020, accepted.

[114] G. Bradski, "The OpenCV Library," *Dr. Dobbs Journal of Software Tools*, 2000.

[115] J. Shi and Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, Jun. 1994.

[116] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker," Microprocessor Research Labs, Intel Corporation, Tech. Rep., 2000.

[117] M. Kieffer and E. Walter, "Guaranteed estimation of the parameters of nonlinear continuous-time models: Contributions of interval analysis," *International Journal of Adaptive Control and Signal Processing*, vol. 25, no. 3, pp. 191–207, Aug. 2010.

[118] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, Nov. 2013.

[119] M. Fanfani, F. Bellavia, and C. Colombo, "Accurate keyframe selection and keypoint tracking for robust visual odometry," *Machine Vision and Applications*, vol. 27, no. 6, pp. 833–844, Jul. 2016.

[120] L. Alvarez, C. A. Castaño, M. García, K. Krissian, L. Mazorra, A. Salgado, and J. Sánchez, "Symmetric Optical Flow," in *Computer Aided Systems Theory – EUROCAST 2007*, Springer Berlin Heidelberg, 2007, pp. 676–683.

[121] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, Sep. 2008.

[122] 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, 2000.

[123] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," *ICRA Workshop on Open Source Software*, vol. 3, no. 2, p. 5, 2009.

[124] V. Drevelle and J. Nicola, "VIBes: A Visualizer for Intervals and Boxes," *Mathematics in Computer Science*, vol. 8, no. 3-4, pp. 563–572, 2014.

[125] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, Sep. 2004.

[126] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[127] E. Mair, M. Fleps, M. Suppa, and D. Burschka, "Spatio-temporal initialization for IMU to camera registration," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Phuket, Thailand, Dec. 2011.

[128] S. Schön, C. Brenner, H. Alkhatib, M. Coenen, H. Dbouk, N. Garcia-Fernandez, C. Fischer, C. Heipke, K. Lohmann, I. Neumann, U. Nguyen, J.-A. Paffenholz, T. Peters, F. Rottensteiner, J. Schachtschneider, M. Sester, L. Sun, S. Vogel, R. Voges, and B. Wagner, "Integrity and Collaboration in Dynamic Sensor Networks," *Sensors*, vol. 18, no. 7, p. 2400, Jul. 2018.

[129] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Aug. 2013.

Publications

Publications as main author

R. Voges, C. S. Wieghardt, and B. Wagner, "Timestamp Offset Determination Between an Actuated Laser Scanner and its Corresponding Motor," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1/W1, pp. 99–106, 2017.

R. Voges, C. S. Wieghardt, and B. Wagner, "Finding Timestamp Offsets for a Multi-Sensor System Using Sensor Observations," *Photogrammetric Engineering & Remote Sensing*, vol. 84, no. 6, pp. 357–366, 2018.

R. Voges and B. Wagner, "RGB-Laser Odometry Under Interval Uncertainty for Guaranteed Localization," in *Book of Abstracts of the 11th Summer Workshop on Interval Methods (SWIM 2018)*, Rostock, Germany, Jul. 2018.

R. Voges and B. Wagner, "Timestamp Offset Calibration for an IMU-Camera System Under Interval Uncertainty," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018.

R. Voges and B. Wagner, "Extrinsic Calibration Between a 3D Laser Scanner and a Camera Under Interval Uncertainty," in *Book of Abstracts of the 12th Summer Workshop on Interval Methods (SWIM 2019)*, Palaiseau, France, Jul. 2019.

R. Voges, B. Wagner, and V. Kreinovich, "Efficient Algorithms for Synchronizing Localization Sensors Under Interval Uncertainty," *Reliable Computing (Interval Computations)*, vol. 27, no. 1, pp. 1–11, 2020.

R. Voges, B. Wagner, and V. Kreinovich, "Odometry under Interval Uncertainty: Towards Optimal Algorithms, with Potential Application to Self-Driving Cars and Mobile Robots," *Reliable Computing (Interval Computations)*, vol. 27, no. 1, pp. 12–20, 2020.

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)*, Las Vegas, NV, USA, Oct. 2020, accepted.

Publications as co-author

S. Schön, C. Brenner, H. Alkhatib, M. Coenen, H. Dbouk, N. Garcia-Fernandez, C. Fischer, C. Heipke, K. Lohmann, I. Neumann, U. Nguyen, J.-A. Paffenholz, T. Peters, F. Rottensteiner, J. Schachtschneider, M. Sester, L. Sun, S. Vogel, R. Voges, and B. Wagner, "Integrity and Collaboration in Dynamic Sensor Networks," *Sensors*, vol. 18, no. 7, p. 2400, Jul. 2018.

About the Author

Personal details

Name	Raphael Voges
Date of birth	November 8th, 1992
Place of birth	Hildesheim, Germany

Education

1999 – 2011	General school education Abitur at Bischöfliches Gymnasium Josephinum, Hildesheim
2011 – 2016	Studies in Computer Science at Leibniz University Hannover B. Sc. and M. Sc. degrees

Scientific work

December 2016 – Present	Research associate at Leibniz University Hannover Institute of Systems Engineering – Real Time Systems Group DFG Research Training Group 2159 (i.c.sens)