Multimodal Sensor Calibration with a Spherical Calibration Target

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M.Sc. Julius Valentin Kümmerle

Tag der mündlichen Prüfung:	18. September 2020
Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr.-Ing. Klaus Dietmayer

Abstract

Automated systems are equipped with an increasing number of diverse sensors to deal with more and more complex tasks. Accurate sensor calibration is an essential prerequisite for fusing information from multiple sensors that enables high reliability in important tasks, such as localization and object detection.

In this work, we present novel methods for multimodal sensor calibration. We introduce a spherical calibration target that can be observed by a wide range of sensors, such as cameras, LiDARs and radars. Our target has multiple favorable characteristics that lead to accurate detections and a compact calibration problem formulation.

Our Euclidean calibration framework is the simplest of three multimodal calibration methods that are presented in this work. The Euclidean calibration framework minimizes Euclidean distances between sphere center observations to find the sensors' poses. The second method, additionally, takes observation uncertainties into account by using a probabilistic formulation of the calibration problem. Our last and most advanced calibration method enables calibrating both, intrinsic sensor parameters and sensor poses.

In our simulation environment and on real data, we evaluate all three proposed calibration methods. We show that even our simplest method provides state of the art results with a mean error of less than 3 mm in translation and $0.1\,^{\circ}$ in rotation for a sensor setup consisting of a camera and a LiDAR. Additionally, our evaluation shows that taking observation uncertainties into account leads to significantly improved results. Finally, we show that the estimation of intrinsic camera parameters can benefits from the complementary measurements of range sensors.

Zusammenfassung

Automatisierte Systeme werden mit einer zunehmenden Anzahl unterschiedlicher Sensoren ausgestattet, um immer komplexere Aufgaben zu bewältigen. Präzise Sensorkalibrierung ist eine essentielle Voraussetzung, um die Informationen der Sensoren zu fusionieren und wichtige Aufgaben wie Lokalisierung und Objekterkennung zuverlässig zu bewerkstelligen.
In dieser Arbeit werden neue Methoden zur multimodalen Sensorkalibrierung vorgestellt. Es wird ein spherischer Kalibrierkörper entwickelt, der von einer Vielzahl an Sensoren, wie beispielsweise Kameras, Laserscannern und Radaren, detektiert werden kann. Der Kalibrierkörper hat mehrere günstige Eigenschaften, welche akkurate Detektionen ermöglichen und zu einer simplen Problemformulierung führen. Es werden drei Kalibriermethoden präsentiert. Die erste Methode minimiert euklidische Distanzen zwischen Beobachtungen des Kugelzentrums, um die Sensorposen zu schätzen. In der nächst fortgeschritteneren Methode werden durch eine probabilistische Problemformulierung zusätzlich Beobachtungsunsicherheiten miteinbezogen. In der dritten Methode werden, neben den Sensorposen, auch intrinsische Sensorparameter geschätzt.
Die drei Methoden werden in Simulation und mittels Realdaten evaluiert. Es wird gezeigt, dass bereits die einfachste Methode sehr gute Ergebnisse liefert, mit einem Durchschnittsfehler von weniger als 3 mm in Position und $0.1\,^{\circ}$ in Orientierung für einen Sensoraufbau mit einer Kamera und einem Laserscanner. Außerdem wird in der Evaluation ersichtlich, dass die Berücksichtigung von Beobachtungsunsicherheiten zu einer deutlichen Verbesserung der Kalibrierergebnisse führt. Schließlich wird experimentell gezeigt, dass die Schätzung der intrinsischen Kameraparameter von den komplementären Messungen der Tiefensensoren profitieren kann.

Acknowledgment

This dissertation was written during my work as a research assistant at the Institute for Measurement and Control Systems (MRT) at the Karlsruhe Institute of Technology (KIT) and at the FZI Research Center for Information Technology. I would like to express my sincere thanks to Prof. Dr.-Ing. Christoph Stiller for the supervision of this work and the creation of the excellent working conditions throughout my 3½ years in his group. Further, I would like to thank my co-supervisor Prof. Dr.-Ing. Klaus Dietmayer for the supervision of my thesis and interest in my work. Additionally, I would like to express my sincere thanks to my colleagues at FZI and MRT for many helpful suggestions, interesting discussions and the fun we had at work. Especially, I would like to thank my good colleague and friend Tilman Kühner for providing key advise many times. Also, I would like to thank the mechanical and electrical workshop for building essential equipment for my experiments.

Karlsruhe, in February 2020 *Julius Kümmerle*

Contents

Notation and Symbols

Acronym

2-D/3-D	**2/3-D**imensional
CDF	**C**umulative **D**istribution Function
DoF	**D**egree **o**f **F**reedom
GLONASS	**GLO**bal **NA**vigation **S**atellite **S**ystem
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
ICP	**I**terative **C**losest **P**oint
IMU	**I**nertial **M**easurement **U**nit
LED	**L**ight **E**mitting **D**iode
LiDAR	**Li**ght **D**etection **A**nd **R**anging
PCA	**P**rincipal **C**omponent **A**nalysis
PDF	**P**robability **D**ensity **F**unction
Radar	**Ra**dio **d**etection **a**nd **r**anging
RANSAC	**RAN**dom **SA**mple **C**onsensus
RCS	**R**adar **C**ross-**S**ection
Slerp	**S**pherical **l**in**e**a**r** inter**p**olation
SVD	**S**ingular **V**alue **D**ecomposition

General Notation

Scalars	Regular lower case	a,b,c
Vectors	Bold lower case	**a**,**b**,**c**
Matrices	Bold upper case	**A**,**B**,**C**

1 Introduction

Automation is one of the key technologies that raises our living standards. It started in the controlled environments of production lines and spread further to public places and even our homes. In the beginning, robots were locked away from humans, whereas today, they are integrated in our daily lives and even interact with us. Autonomous mobile platforms guide us through museums, autonomous vacuum cleaners scan and navigate through our homes, modern cars perceive and analyze their environment to make driving more secure and comfortable for us. Robots are capable of completing increasingly complex tasks in unstructured environments and achieve increased reliability which is why we trust these systems more and more.

There are multiple technological developments that brought robots closer to humans. On the one hand, improvements in methods for e.g. localization and perception enabled robots to better understand and interact with their environment. On the other hand, hardware innovations lead to a significant increase in computational power and new sensors provide more accurate, diverse and dense data. The combination of innovation in software and hardware enabled robots to become our assistants for many tasks.

1.1 Motivation

The boost in computational power enabled to process significantly more sensor data simultaneously. That is why robots, such as autonomous cars, can be equipped with more and more sensors to scan the environment. For over a decade, series cars are equipped with multiple radars. In todays high-end series cars, a multi camera setup provides 360° surround view for the driver. To make driver assistance more intelligent, LiDARs start to get into series cars. In fully autonomous experimental vehicles, 3D LiDARs are heavily used for tasks like localization and object detection. In combination with data from cameras and radars, many experimental vehicles are capable of driving safely in many scenarios.

To fuse data from multiple sensors, it has to be transformed to a common reference coordinate system. This is only possible if intrinsic sensor parameters and the relative position and orientation of the sensors to each other are known. The process of finding these parameters is called sensor calibration.

Sensor calibration is an increasingly important topic because more sensors are mounted on robots and the demands on accuracy are raised due to the growing complexity of the tasks. Additionally, more diverse sensors are combined in a setup. Diversity of the sensors adds complexity to the calibration problem because of different measurement principles and sensor characteristics. There are well understood standard methods for camera calibration, however, multimodal calibration for cameras, LiDARs and radars is less researched. Often, calibration of a multimodal sensor setup is split up into multiple calibration procedures which is time consuming and suboptimal for accuracy from a theoretical point of view. This motivates the scope of this work.

1.2 Scope

We present a calibration framework for cameras and range sensors like 3D LiDARs, 2D LiDARs and radars. The number and combination of sensors is arbitrary. Our calibration method allows for estimating the relative poses of the sensors and internal sensor parameters. The calibration accuracy, which we aim for, complies with the accuracy needed for state of the art sensor fusion methods used for e.g. autonomous driving. We use a dedicated spherical calibration target which is moved around the sensor setup. The calibration process can take place indoors and outdoors. The process is highly automated. Only the calibration target has to be moved manually around the sensor setup. The complete calibration process takes between 1 min and 10 min depending on the number and types of sensors.

1.3 Contributions

Contributions of this work are summarized in the following:

- A spherical calibration target that can be accurately detected in camera, LiDAR and radar data is designed. The spherical shape has many benefits such as view invariance and ease of modeling. Further, it has practical advantages such as the Styrofoam sphere being a cheap off-the-shelf product and its light weight. We present target detectors for cameras and range sensors that achieve sub-resolution detection accuracy.
- An extrinsic calibration framework that provides live feedback and calibration results on runtime is developed. Visualizations and quantitative checks allow the user to assess the quality of the calibration while recording and to decide whether more data is needed or not. This simplifies and speeds up the calibration process significantly.
- We propose a probabilistic formulation of the calibration problem. This allows to incorporate observation and measurement characteristics of the sensors in the calibration problem. We show that this significantly improves calibration quality compared to ignoring the different characteristics.
- We present a joint calibration framework that simultaneously calibrates extrinsic and intrinsic parameters of a multimodal sensor setup. Individual noise characteristics of every individual measurement is taken into account. Measurements are integrated continuously by using a spline to model the target trajectory. We show that camera calibration can benefit from additional information of range sensors.
- We propose a simple but effective method to calibrate a sensor setup relative to keypoints such as the rear axle of a car or an antenna. By temporarily extending the sensor setup with additional sensors, the keypoints can be observed. The position of the keypoints relative to the main sensor setup can be derived by calibrating the main sensor setup to the temporarily added sensors. We show the practicability of the method in our evaluation.

1.4 Outline

The outline of this work is as following:
We begin with an overview of the current state of the art in calibration of cameras, LiDARs and radars (chapter 2). First, we summarize methods that use calibration targets. We survey methods that only calibrate cameras, then we look at approaches for camera to LiDAR calibration, and finally, we also look at publications on radar calibration. Next, calibration methods without a dedicated calibration target are summarized. We finish our literature survey with the topic of measurement noise in the context of multimodal calibration.
In chapter 3, we discuss the fundamentals that are used in our calibration methods. We cover the topics sensors, computer vision, statistics, 3D geometry and model fitting. Fundamentals are only presented in chapter 3 to keep the chapters in which our calibration methods are presented more compact and clean. Experienced readers can simply skip the fundamentals and dive right into the following methodology chapters.
In chapter 4, we introduce our calibration target. We define criteria that the calibration target should fulfill. Construction details are given to easily replicate the target. In chapter 5, we introduce our target detectors for cameras, LiDARs and radars.
Our *Euclidean calibration framework* is presented in chapter 6. It solves the calibration problem by minimizing Euclidean point-to-point, point-to-ray and ray-to-ray distances between sphere center observations. Linear interpolation is used to generate time-synchronized target observation pairs to link multiple sensors.
The advancement of the Euclidean framework is our *probabilistic calibration framework* which is presented in chapter 7. It uses a probabilistic formulation of the calibration problem to incorporate different observation characteristics of the sensors. A universal sensor model is introduced which leads to a simple and efficient solution.
In chapter 8, we present our most advanced calibration framework which we call *joint calibration framework*. Instead of using sphere center observations, it takes raw measurements as input data. This allows to estimate extrinsic and, additionally, intrinsic sensor parameters.
In chapter 9, we present a simple but effective method to calibrate a sensor setup relative to keypoints. A step-by-step instruction for calibrating a sensor setup to the rear axle of a car is given. This chapter marks the end of the

methodology part.
An evaluation of the presented methods is given in chapter 10. We use simulated and real data to evaluate the target detectors and compare calibration results of our different calibration frameworks in multiple experiments.
Chapter 11 provides a conclusion and proposes several promising extensions to our work.

2 State of the Art

Sensor calibration is a well researched topic. A large amount of methods is published in literature. In the following, we summarize the most important calibration methods.

2.1 Calibration with a Target

In this chapter, we provide a survey on calibration methods that make use of a calibration target.

2.1.1 Camera Calibration with a Target

For almost half a century, camera calibration is an active research topic in e.g. the photogrammetry and the computer vision community. Hence, a huge amount of methods has been published in literature. Since our work is more focused on multimodal sensor calibration than camera calibration, we only discuss a small part of state of the art camera calibration that is relevant for our work.

The approach presented in [Zha00] represents a milestone of camera calibration. The authors propose to calibrate a camera by using a single planar board with printed squares of known size and relative position on it. The novelty is the simplicity of the target. It is easy and cheap to build, and therefore, is available to almost everyone. Previously published methods need more complex targets which makes calibration more effortful. Multiple views on the target are captured by moving the board or the camera. The corners of the printed squares are detected and the camera parameters are estimated by solving a minimization problem over the reprojection error. The authors report

state of the art results while keeping costs for equipment very low. Others proposed to use circle prints [Hei00, KB06] but squares of a checkerboard pattern are considered the standard today because of their high detection accuracy.
This simple calibration procedure is evaluated on a pinhole model in [Zha00] and is applied to more complex camera models in e.g. [KB06, SMS06, BS18]. The resulting accuracy is sufficient for common tasks like image undistortion and 3D reconstruction [SMS06, Str15].

Besides the popular planar calibration boards, there are also 3D calibration targets, one of which is a sphere. In [DDL94], a single camera is calibrated by observing multiple spheres. A simple pinhole camera model with principal point, skew and focal length is used. The projection of a sphere on a pinhole model is an ellipse. In a first step, the principal point and skew is estimated. Then, the focal length can be derived from a single sphere. The authors report high errors on the focal length estimation.
In the work [TX02], a more theoretical view on camera calibration with spheres is presented. By using the concept of absolute conics, the authors show that each sphere of unknown size provides five constraints, three of which are needed to estimate the relative size and position of the sphere. So, two constraints can be used to infer the camera intrinsics. The authors use a minimum of three spheres to calibrate their pinhole camera model with five parameters. They solve for all parameters at once by minimizing the reprojection error.
Since the minimization needs a good initialization, an efficient closed-form solution based on an algebraic error is proposed in [AD03]. The authors also extend the approach to multi-camera setups. They calibrate intrinsic and extrinsic parameters of all cameras at the same time. They highlight the advantage that a sphere can be observed from every position which is not possible for planar targets. However, they recognize problems for their solving procedure when an ellipse degenerates to a circle. Further, they notice that the calibration quality highly depends on the accuracy of edge detections.

Compared to a checkerboard, a spherical calibration target has some downsides for camera calibration. We will later see that these downsides can be compensated in a multimodal calibration method and a sphere even shows potential to improve camera calibration.

2.1.2 Camera-LiDAR Calibration with a Target

Calibration of cameras and LiDARs is usually performed with a dedicated calibration target. In contrast to camera calibration, there is no standard target for camera-LiDAR calibration which is used by a majority. Instead, a great variety of targets were introduced in literature in the last two decades.

The earliest published methods on camera-LiDAR calibration are designed for a camera and a 2D LiDAR (measurements only in one plane).
The method introduced in [ZP04] is one of the first that addresses the problem of calibrating a camera and a 2D LiDAR. The authors propose to use a checkerboard that is observed from different views. They use an iterative solving scheme that estimates the poses of the checkerboard and the pose difference between the sensors. The final solution is based on minimizing the camera reprojection error and the point-to-plane error for the LiDAR data. These two cost terms cannot be directly compared to each other. Therefore, a scaling factor that has to be tuned by the user is introduced. The authors report high influence of the scaling factor and cannot provide a rule to find a good choice. This makes the method unreliable.
With the goal to provide a more reliable calibration method, the approach presented in [LLD$^+$07] uses another cost term. Again, a 2D LiDAR is calibrated to a single camera. The target is a planar board that is shaped like an arrow (see Figure 2.1 b). The edges of the target are detected in the camera images and the LiDAR points that are closest to the edges are segmented. These points are then projected to the image and the point-to-line errors are minimized for multiple views of the target. No weighting factor is needed as in the previous method because only one type of cost term is used. Direct comparison to [ZP04] shows increased reliability and improved accuracy in orientation.
Neither of both previously discussed approaches is fully automatic. The user has to segment edge points in the LiDAR data and corresponding edges in the camera images. [KP10] proposes a fully automatic detection pipeline and an improved checkerboard corner detector. This results in improved convenience and accuracy compared to the previously discussed methods.
The authors of [VBN12] use the same approach as [ZP04]. A checkerboard is observed from different views and a final calibration is calculated by minimizing the reprojection error and the point-to-plane error which is scaled by a

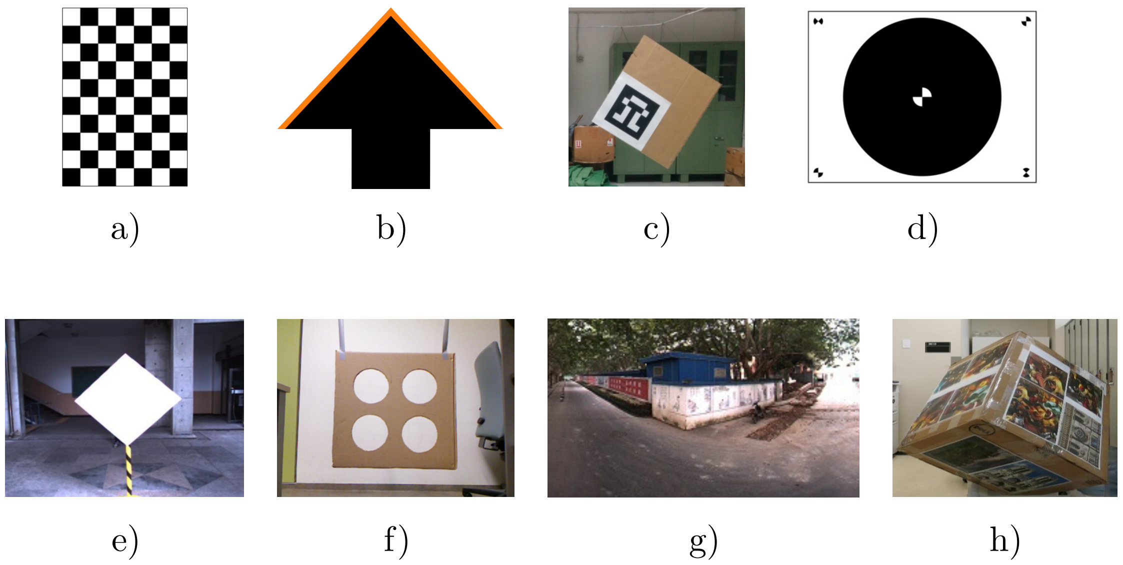


Figure 2.1: Different targets used for camera-LiDAR calibration in literature. a) A standard checkerboard such as the one used for camera calibration. b) An arrow shape. The orange edges are detected and matched in camera and LiDAR. c) A cardboard with marker [DCRK17]. d) A planar board with circle and markers [ABB12]. e) Planar board of precisely known dimensions [PYW$^+$14]. f) Cardboard with four holes [VSMH14]. g) Corner of a wall that forms a trihedron with the ground plane [GLL13]. h) Cardboard box with texture printing [HMES16].

manually tuned weighting factor. The contribution of [VBN12] lies in the very efficient generation of a few hypotheses that serve as strong initial solutions. This makes the calibration process more efficient and robust compared to the original method.
To summarize, for calibrating a camera and a 2D LiDAR, usually, a planar target is used and most approaches minimize a combination of reprojection and point-to-plane errors.

For calibrating a 3D LiDAR, the diversity of targets is high. As for 2D LiDARs, also planar targets are popular.
[GMCS12] uses multiple checkerboards which are distributed in the environment. With a single scan of the static scene, multiple cameras and a LiDAR can be calibrated. The sensor setup is calibrated sequentially, starting with standard intrinsic and extrinsic calibration of the cameras which is based on minimizing a reprojection error. Then the LiDAR is calibrated to the cameras. The 3D positions of the checkerboard corners are determined from the cameras, and the LiDAR points hitting the checkerboards are segmented. The

final solution is calculated based on minimizing point-to-point distances of the LiDAR and camera point clouds with the popular iterative closest point (ICP) algorithm. Since both point clouds are sparse, ICP does not reliably provide high accuracy results for the pose difference between camera and LiDAR.
In [DCRK17], a rectangular cardboard with a marker printed on it is used (see Figure 2.1 c). The corner points of the cardboard are detected in camera and LiDAR and the calibration problem is solved by minimizing 3D point-to-point distances of the corner correspondences. The authors argue that this is more accurate than the method discussed before since their point pairs represent the same object points.
The authors of [ABB12] also use a planar board but they print a circle on it with a corner marker that is placed on the circle center (see Figure 2.1 d). They detect the board in the 3D point cloud of the LiDAR and the circle which appears as an ellipse in the camera image. From the ellipse and the corner marker, the 3D position of the circle center can be derived. The calibration problem is formulated as a minimization problem of point-to-plane distances of the circle center from camera to the plane from LiDAR for multiple views.

The previously discussed approaches all use a pattern or marker on a planar board. Another approach is to only use the shape of the board such as the authors of [PYW$^+$14]. They use a polygonal board of precisely known dimensions (see Figure 2.1 e) and detect the 3D corner points in the LiDAR point cloud and 2D corners in the image. They minimize the reprojection error of the corners for multiple views.
Some methods use circular holes in a target. A single circular hole in a board is used in [RFFB08]. Around this hole, a circle is drawn so that in camera and LiDAR data the 3D center of the hole can be determined. Point-to-point distances are minimized as an initial solution which is then refined by also taking the orientation of the target into account. Since the target is comparably small, the 3D position estimation in LiDAR data is not very accurate. Therefore, the calibration results of this early method are not reliable.
The authors of [VSMH14] use a planar board with four circular holes (see Figure 2.1 f). They estimate the 3D circle centers as well and calculate an initial calibration result in a single shot. The authors recognize that the calibration quality based on holes is not high enough. For refinement, they take all edges in the scene into account and minimize the reprojection error such as in [LT13]. For optimizing over all edges, a good initial solution is essential due to many local minima.

In [DKG19], the same planar board with four circular holes is used but the target is significantly larger. They also minimize point-to-point distances of corresponding circle centers. Because of the larger target, the circle based calibration is more accurate than the one in [VSMH14].

Planar calibration targets are not ideal for 3D LiDAR data. The problem is that the target cannot be localized with high accuracy. The position is estimated by using edge points but an edge is in between a point on the target and a point in the background, so, the exact edge location cannot be measured. The more scan lines a LiDAR has, the more edge discontinuities can be measured and the better the estimate becomes due to averaging. But still, many people recognized this problem and decided to use a 3D calibration target that can be well localized in 3D point clouds.
The authors of [GLL13] use an arbitrary trihedron to calibrate a camera and a 3D LiDAR (see Figure 2.1 g). The three planes can be in arbitrary angles to each other, as long as they are not coplanar. They proposed to use a house corner with some texture on it. They move the sensors around the trihedron and manually segment it in the point clouds and the images. For each view, the full pose of the LiDAR is estimated relative to the trihedron based on fitting three planes. Salient points are detected and matched over all camera images and the camera motion is estimated. The calibration problem is formulated as a minimization problem with cost terms enforcing planarity and motion consistency. An advantage of this method is that the calibration target can be very large which theoretically allows for accurate pose and motion estimation. The downside of using a house corner is that it might not be exactly planar.
In [HMES16], a cardboard box with texture prints on it is used to calibrate a stereo camera and a 3D LiDAR (see Figure 2.1 h). By detecting the structure on the box and matching it from one to the other camera, a 3D point cloud is created with the precalibrated stereo camera. This pointlcoud is aligned to the point cloud from the LiDAR by using ICP. This method cannot be used with arbitrary texture to calibrate a mono camera and a LiDAR. In this case, a known checker pattern could be used.
Also the authors of [PEH18] use a cardboard box but without texture. A single camera and a 3D LiDAR are calibrated by estimating three planes in the LiDAR data and the corresponding corner point in the image. User input is needed to segment the appropriate regions in the 3D point cloud and the image. The link between LiDAR and camera is established by minimizing the distance between projected 3D corner points and 2D corners in the camera image. A

problem of this approach is that the corners are not precisely detectable in the camera image. Further, a lot of user input is needed which is inconvenient.

This survey shows, how diverse the calibration targets and the problem formulations for camera-LiDAR calibration are. An important lesson to learn from this survey is that the classical checkerboard is not ideal for calibrating 3D range sensors. 3D targets can be more accurately detected in 3D range data.

2.1.3 Radar Calibration with a Target

Radars are very common sensors in the automobile industry and since cameras and LiDARs started to get into series cars, the calibration of radars to cameras and LiDARs became increasingly important. Yet, in scientific literature, radar calibration is not very popular. In the following, we summarize the most popular and important approaches.

In [STTO04], a radar is calibrated to a camera. The authors use a metal corner reflector which is well visible in radar data. The reflector is moved up and down orthogonally to the scanning plane of the radar to generate a signature in the radar cross-section signal that is used to segment the calibration target from the rest of the reflecting objects in the environment. Template matching is used to detect the corner reflector in images. The accuracy of such detections is questionable. Instead of directly estimating the relative rotation and translation between the two sensors, the authors estimate the homography between image plane and radar scanning plane. A qualitative evaluation shows decent results for this early approach.
The authors of [WZXM11] also calibrate a radar to a camera but this time with a rectangular metal panel. The only benefit of a metal panel compared to a metal corner reflector is that it is easier to build. The downside is that the strength of the reflected signal highly depends on the orientation of the panel which makes the calibration process more cumbersome. Again, the homography between radar scanning plane and image plane is estimated.
For the method presented in [NAAR$^+$15], multiple corner reflectors are placed next to each other. The authors present two approaches for radar and camera calibration. For the first approach, the distances between the reflectors must be known. For the second approach, the reflectors are only assumed to be fixed and the exact distances do not need to be known. In this case, the sensor

setup is moved around the targets to observe the scene from multiple views so that the distances between the reflectors can be estimated precisely. The detection of the corner reflectors is not automatic. In radar and camera data, the corners of the reflectors have to be marked manually, whereas a Harris corner detector supports the user in images. Further, because multiple targets are used simultaneously, the user has to determine the correct associations between target detections in radar and in camera data.
Up to now, all discussed methods are used to calibrate a radar to a camera. [PMP19] proposes a calibration target that allows to calibrate a radar, a camera and a LiDAR. The target is build from a Styrofoam triangle on which a checkerboard pattern is printed. A corner reflector is attached in the center of mass of the triangle. A camera can estimate the pose of the target based on the checkerboard pattern. The triangular shape is detected in LiDAR data. The corner reflector is needed to locate the target in radar data. To refine the estimated scanning plane of the radar, the authors assume that the radar cross-section of a measurement is influenced only by the elevation angle to the scanning plane of the radar. A quadratic curve is fitted to measured pairs of radar cross-section and elevation angle to define a characteristic curve. The scanning plane of the radar is then refined by minimizing the difference of the measured radar cross-sections to the values resulting from the characteristic curve. The assumption that the radar cross-section only depends on the elevation angle of the measurement is not correct in practice. E.g. the orientation of the corner reflector influences the radar cross-section as well. That is why determining the elevation angle only based on the radar cross-section is very noisy.
One of the core problems of calibrating radars to cameras and LiDARs is that the observation accuracy of a radar is significantly lower than for the other sensors. The previously discussed approaches ignore the different observation accuracies. The authors of [DKG19] propose to use the Mahalanobis distances between observations. By minimizing these distances, the poses and covariances are optimized. The covariances ideally represent the different observation accuracies of the sensors. Estimating the covariances simultaneously to the poses can lead to problems since by varying the covariances also the sets of outliers and inliers are redefined. This often leads to unreliable optimization results and the covariances do not represent the real observation accuracies. Prior knowledge of the observation noise improves reliability significantly.

2.2 Targetless Calibration

Sensor calibration without a dedicated calibration target is compelling because of many reasons. Depending on the calibration target, it may take some time to construct and build it. Some calibration targets are heavy or very large so that transportation is not easy. In this case, calibration is only possible at the location of the immobile target. Further, targetless calibration allows for calibration during operation which is not possible with a target. This is desirable because calibration parameters often can change over time due to forces on the sensor platform, temperature changes or aging. For some sensor setups, changes during operation are applied on purpose such as for zoom lenses or sensors with controlled tilting.
In the following, we review existing approaches for targetless camera and camera-LiDAR calibration.

2.2.1 Targetless Camera Calibration

Calibrating cameras without a target is a research topic for many years. However, the number and especially the diversity of publications is significantly lower than for targetless calibration of cameras and LiDARs.

In the work [BMRS12], a single camera is calibrated without a target. A pinhole model with radial and tangential distortion is used. The camera must observe a scene from different orientations and positions. It is essential that the scene is static, rich in texture and does not contain symmetric patterns. Salient landmarks are extracted from the scene and matched in each image. The reprojection error of these landmarks is minimized by estimating the camera pose for each image and the intrinsic camera parameters. This is possible only up to the scale.
In [RKBL17], the intrinsic and extrinsic parameters of a stereo camera are estimated. Bundle adjustment is used to optimize 3D positions of landmarks, sensor movement, extrinsic parameters and intrinsic parameters up to the scale. The novelty of this work is to split up the expensive calculations of bundle adjustment into several stages which separately can be computed in real time. This enables calibration during operation.
The authors of [DHS09] also use salient landmarks for camera calibration.

Their setup consist of two cameras. The authors propose to use a recursive optimization schema (iterated extended Kalman filter) to continuously refine the calibration with low computational costs. An epipolar criteria is developed that does not rely on temporal matching of landmarks. This leads to a robust calibration even if objects are moving in the scene.
In [MW16], the extrinsic parameters of a camera pair are calibrated. Additional IMU measurements are used to infer the scale. Again, landmarks are detected and tracked. An Unscented Kalman Filter is used to iteratively optimize landmark positions and the sensor poses. The downside of this approach is that substantial angular rotation around two axis is required. This is often not possible for on-road vehicles.

Targetless camera calibration suffers in general from multiple problems. Without additional information, cameras can only be calibrated up to the scale. The most accurate methods are based on bundle adjustment which is highly dependent on the amount of static salient points in the scene. Often, to make the calibration more stable, not all parameters are calibrated with a targetless method. E.g. the calibration problem is constrained by given sensor positions or tangential distortion parameters.

2.2.2 Targetless Camera-LiDAR Calibration

In literature, many approaches for calibrating cameras and LiDARs without a target are presented. They differ significantly in their underlying principles, degree of automation, practicability and computational expense.

A very common approach that became popular with the work [VW97] is based on *mutual information*. Mutual information is a concept of information theory. It is a measure of the dependency of two random variables. More precisely, the mutual information $I(X;Y)$ measures how much information of the random variable X can be inferred by observing the random variable Y.
In [VW97], mutual information is used to register multimodal images. E.g. magnetic resonance images and computed tomography images are aligned by maximizing the mutual information between the data. This concept is the basis for many publications about targetless camera and LiDAR calibration.
In [MKF09], a setup for collecting aerial data consisting of a camera and a terrestrial LiDAR are calibrated based on mutual information. An image with

elevation and reflectance channel is generated from the LiDAR data. The extrinsic parameters of the sensor setup are then optimized based on a mutual information score function. The authors recognize that the depth discontinuities (or elevation jumps in aerial data) are highly correlated with changes in the visual image. The reflectance information provides additional useful information but only leads to minor improvement.
The authors of [PMSE12] present a mutual information approach that only uses LiDAR reflectivity and camera images. They show by experiment that in many scenes there is a high correlation between the intensities of a camera image and the reflectivity data of a LiDAR. Further, they show that, for a sufficient number of views, the sample variance of the estimated parameters empirically converges to the Cramer-Rao-Lower-Bound. They infer that their method is a minimum variance unbiased estimator. Experiments with real data show limitations of the approach. The authors notice that a high amount of edges in the camera image that do not correspond to edges in the reflectivity image from the LiDAR cause a degradation of the estimation quality. One type of such misleading edges is large shadows that appear in the camera image but are not present in the reflectivity data of the LiDAR.
An insightful benchmarking of the just mentioned approach is presented in [ON15]. The approach is compared to a method that uses a dedicated calibration target. Based on real data, they show that the targetless method cannot compete against the method with a calibration target in terms of accuracy. They explain that outlier detection and rejection is simpler and more reliable with a known target than in an unknown environment. This is the reason why the mutual information approach is less robust and, on average, less accurate. They conclude that, at least for high precision applications such as ground surveying, the approach based on mutual information is not sufficient.

A closely related concept to mutual information is χ^2 statistics. In [BPDA00], the authors use a χ^2 similarity measure to align depth maps from LiDAR with camera images to register the sensors. They report that the registration quality with their χ^2 similarity measure is comparable to the approach with mutual information. They report a great number of local minima that makes finding the global minimum hard. This is also a problem when using mutual information. As a conclusion, the authors claim that the mutual information measure, which is considered to be a standard, can be substituted by other information theoretic similarity measures.
Also in [WLH+04], the authors use the χ^2 similarity measure to calibrate a

camera and a LiDAR. They not only use depth but also reflectivity information from the LiDAR. The 3D LiDAR data is rendered to a 2D image with depth and reflectivity channels. As with mutual information, they also report that registration from arbitrary misalignments is not possible due to many local minima. So, they also have to use initialization routines to already start at a good solution.

The previously discussed methods for targetless calibration of camera and LiDAR are global methods which use the complete scene. Another approach is to focus on local features.
The authors of [SA01] propose to use rectangular structures which are identified and matched in the camera and 3D LiDAR data. They mainly focus on urban scenes in which these rectangular structures often occur. They initialize rotation by extracting vanishing points. 2D lines that intersect in a vanishing point are matched to parallel 3D lines. The matching of rectangular structures in images and 3D data is based on a sampling-based approach. The final calibration is calculated based on comparing 3D lines which are projected onto the camera image and 2D lines which are detected in the camera image. The exact matching score is not given by the authors.
In the work [TK13], a method that calibrates a camera and a LiDAR based on arbitrary planar structures is presented. The 3D planar structures from LiDAR are projected into the camera image and an alignment cost is minimized. The authors compare their method to mutual information based approaches. They report that the accuracy is in between mutual information approaches that use only depth information and approaches that additionally use reflectivity information. One drawback of this method is that it is not fully automatic. The user has to identify planar regions in the camera image and match them with the corresponding region in the 3D data.
Also the authors of [SHS07] present a method that needs user input to identify features in a scene. The main contribution is a special visualization of range data which allows the user to simply match corresponding points in LiDAR and camera. An advantage of this method is that it does not rely on the presence of special features such as planar regions and is therefore applicable in a wide range of scenes.
A fully automatic approach is introduced in [Bil09]. A camera and a LiDAR are calibrated based on a multi-stage process consisting of multiple initialization steps and a final optimization stage. For the final stage, points at depth discontinuities are projected to the camera image. These points are associated

to close edges and the according distances are minimized. The method heavily relies on a good initialization due to possibly wrong associations with a bad starting configuration. The authors report that the approach is not as stable as methods which use a known calibration target.
The method introduced in [LT13] is also based on optimizing the distance between projected depth discontinuities and edges in the image. The main idea is to start with a very good calibration which is determined based on a reliable method with a known target. The proposed framework can detect and correct for changes in the sensor poses during operation. Due to a very good initialization, the optimization can be bounded to a small range in which usually only one minimum, the global minimum, exists. The authors argue that this is the reason why their method is very reliable.
In the work [CKB16], another method that is based on aligning depth discontinuities in 3D data with 2D edges in an image is presented. The authors combine calibration with depth upsampling. The motivation behind this is to extend the alignment measure with the additional upsampling to create a stronger similarity measure. They optimize upsampling and calibration based on one common cost function. The authors argue that the upsampling process provides useful information to the calibration and vise versa so that both the upsampling and the calibration are improved.
Another feature-based approach is presented in [NKB19]. Thin objects like poles and traffic signs are used as features. In contrast to the other previously explained methods, the costs are not calculated in the 2D image space but in 3D. A sequence of images is used to generate a point cloud based on structure from motion. The same feature extractor is used on the point cloud from camera and the one from LiDAR. The features are aligned in 3D by optimization of the sensor poses.

After discussing global methods (mutual information and χ^2 similarity) and local feature-based approaches (planar regions, edges, thin objects, ...), we look at methods that use motion as another information that can be deduced from both camera and LiDAR data.
In [AKU$^+$06], the authors present a method to calibrate a camera and a LiDAR based on the motion of detected objects. The objects can be arbitrary but should not be occluded. The similarity of the objects' motion is scored by mutual information.
The authors of [IOI18] present a method that also registers a camera and a LiDAR based on motion. The difference to the previous approach is that the

motion of the sensors is used instead of the motion of other objects. The motion of the LiDAR is estimated based on ICP. Camera motion is derived by tracking feature points in consecutive images. Since a monocular camera is used, the scale of the motion cannot be measured directly. In an iterative process, the scale of the camera motion is derived from the 3D data of the LiDAR. 3D points are projected into the image and tracked in both domains. By minimizing the reprojection error, the scale and the calibration parameters can be estimated. One major disadvantage of this method is that the sensors have to be rotated around all axes. This is hard for wheeled platforms such as autonomous cars, as steep hills would be needed.

We conclude that there are mainly three different approaches in literature for extrinsic calibration of camera and LiDAR without a target. Global approaches that do not use specific features are usually fully automatic but suffer from many local minima which makes a strong initialization essential. Feature-based methods suffer from the same issue of local minima but can be robustified by user input to resolve the association problem. We also discussed motion as multimodal information. Approaches of this type are usually fully automatic but also suffer from assumptions such as objects not being occluded (if motion of objects is used) or rotation about all axes (if sensor motion is used).
Each targetless approach that is compared to a method which uses a calibration target is reported to be less accurate and less robust than this reference method. This is the reason why, in practice, targetless methods are only used to monitor the quality of the calibration or correct for small changes.

2.3 Measurement Noise

Most calibration methods neglect measurement noise completely. In this case, all measurements equally influence the calibration result even if they have significantly different noise distributions. For calibration of a single sensor type, ignoring the measurement characteristics can be a reasonable simplification because the measurements can be modeled similarly. This is not the case for sensors of different types. E.g. the range noise of a 3D point measured with a LiDAR is almost independent of the distance of the point but for a stereo camera the range noise increases with distance. Neglecting such characteristics in

multimodal calibration leads to suboptimal results.
The influence of weighting measurements from different sensor types is discussed in [ZP04]. A simple weighting factor is introduced to compare measurements from a camera and a LiDAR. The authors report that the weighting factor has a major influence on the quality and reliability of their calibration method.
Also in [ZD12], the problem of varying measurement noise is recognized. A camera and a LiDAR are calibrated with a checkerboard. The authors argue that for each view on the checkerboard the detection accuracy in the camera is different because of e.g. the viewing angle. In contrast to this, they recognize that the observations from a LiDAR show less variance in accuracy. So, their method uses estimated covariances of the camera observations to weight costs from different views in their optimization problem. Still, their approach is not fully consistent since their weighting is only based on the observation characteristics of the camera and LiDAR measurement noise is neglected.
To the best of our knowledge, there is no multimodal calibration method presented in literature that fully and consistently propagates the measurement characteristics through the complete problem.

3 Fundamentals

In this chapter, we present fundamental concepts and techniques that are used in our calibration methods. We first discuss the working principles of common sensors in robotics. Next, we present computer vision tools that are later used to detect the calibration target in images. Then, statistical tools are discussed. They will be used for a probabilistic formulation of the calibration problem and for error propagation. Up next, concepts of 3D geometry are summarized. This chapter is concluded with the topic of model fitting which is essential to solve the formulated calibration problem.

3.1 Sensors

In the following, we introduce the most relevant sensor types for many robotic applications such as automated driving. We discuss their functional principles and how to model them.

3.1.1 Camera

Cameras provide a lot of useful information about the environment. The human eye proofs that the visual information from cameras is even sufficient to drive a car safely. Cameras are small and cheap which makes them popular for many applications.

Cameras can be represented as a photosensitive sensor and a lens. Light is bundled by the lens and directed to the sensor. The three dimensional environment is then represented as a two dimensional projection on the sensor. A *camera model* is used to model this projection.

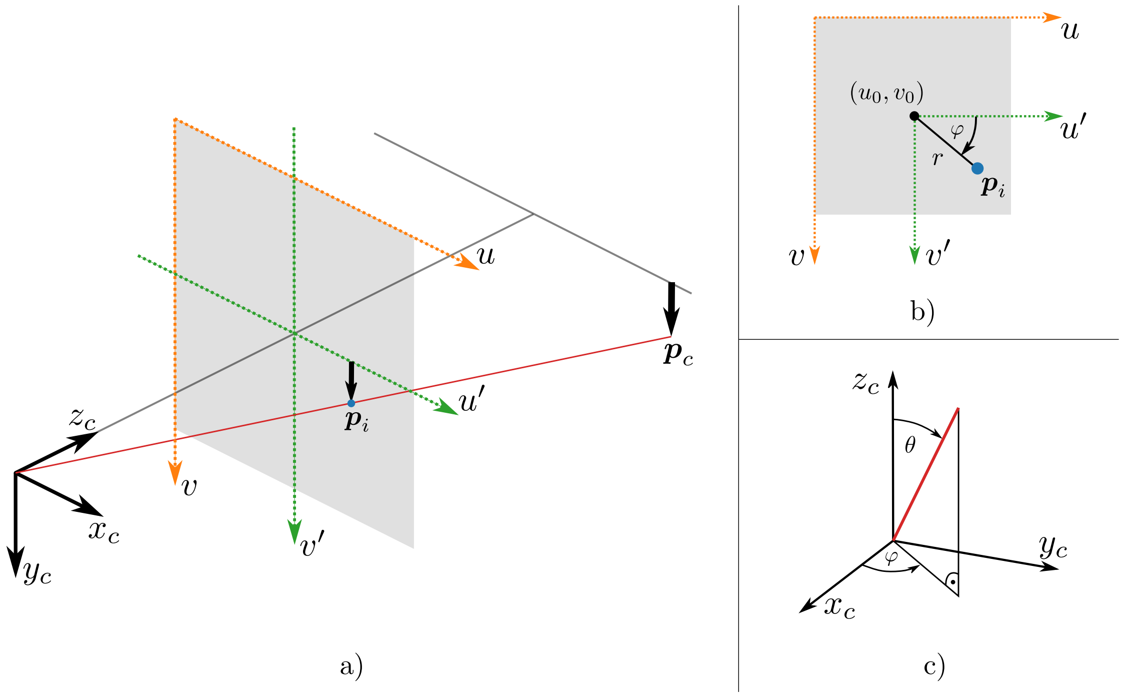


Figure 3.1: Visualization of different coordinate systems for the camera model.
a) A 3D scene point $\boldsymbol{p}_c$ is projected onto the sensor (image plane). The projected point is denoted as $\boldsymbol{p}_i$ (blue) and appears in the image coordinate system (u, v) (orange). We introduce a shifted and scaled image coordinate system (u', v') (green) to explain the camera model. To express the 3D viewing ray (red) that corresponds to the projected point $\boldsymbol{p}_i$, we introduce the camera coordinate system (x_c, y_c, z_c).
b) The point $\boldsymbol{p}_i$ can be expressed in polar coordinates (r, φ).
c) A ray in camera coordinate system described by the inclination angle θ and the azimuthal angle φ.

More formally, a camera model is a mapping between a 3D scene point $\boldsymbol{p}_c$ in camera coordinates and a 2D image point $\boldsymbol{p}_i$ in image coordinates (see Figure 3.1). Only in special cases, this mapping is invertible. Usually, we have to decide which direction of the mapping should be solvable in closed form. A forward model $\mathcal{P}_f : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ defines the mapping from a 3D scene point to a 2D image point. A backward model $\mathcal{P}_b : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ defines the 3D viewing ray that corresponds to a given image point. Depending on the application, it is more important to work with image points (use a forward model) or with viewing rays (use a backward model).

When searching for a suitable camera model, the most important factor is the lens. There are different camera models which are commonly used for special types of lenses. In the following, we summarize a modular backward camera model presented in [Str15] that allows to handle a wide range of lenses: Let (u, v) be a point in the image space. First, this point is centered by the *principal point* (u_0, v_0) and normalized by the *focal length* f:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \frac{1}{f} \begin{pmatrix} u - u_0 \\ v - v_0 \end{pmatrix} \quad . \tag{3.1}$$

The principal point is usually close to the center of the sensor and represents the rotation center in case of a rotation symmetric mapping. The focal length is a measure of how strongly the lens converges rays of light and can be interpreted as the distance in which the rays, coming from infinity, are focused in the optical center. Normalizing with the focal length is used to improve numerical stability when estimating a distortion model.
Our distortion model differentiates between radial and tangential distortion. Tangential distortion usually is due to errors in manufacturing and assembly. Radial distortion models distortion effects especially of small lenses or wide angle lenses. Mathematically, we use the following distortion model:

$$\begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = \begin{pmatrix} u'(k_1 r^2 + k_2 r^4 + \ldots) + 2p_1 u'v' + p_2(r^2 + 2u'^2) \\ v'(k_1 r^2 + k_2 r^4 + \ldots) + p_1(r^2 + 2v'^2) + 2p_2 u'v' \end{pmatrix} \tag{3.2}$$

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \begin{pmatrix} u' + \Delta u \\ v' + \Delta v \end{pmatrix} \quad , \tag{3.3}$$

with $r^2 = u'^2 + v'^2$, the distortion difference $(\Delta u, \Delta v)$ and the undistorted point (u_d, v_d). In Equation 3.2, the first terms $u'(k_1 r^2 + k_2 r^4 + \ldots)$ and $v'(k_1 r^2 + k_2 r^4 + \ldots)$ remove radial distortion. The other terms remove tangential distortion.
We use a *projection model* to describe the main rotation symmetric characteristic of the lens. If the projection model suits the lens, the radial distortion is small and can be described with only a few parameters. Mathematically, we define a projection model as a function $\mathcal{A}$ that maps from the inclination

angle θ of the ray (see Figure 3.1) to the (normed) point radius. The following projection models are commonly used in lens design:

$$\text{Projective:} \quad \mathcal{A}(\theta) = \tan(\theta) \tag{3.4}$$

$$\text{Stereographic:} \quad \mathcal{A}(\theta) = 2\tan\left(\frac{\theta}{2}\right) \tag{3.5}$$

$$\text{Equiangular:} \quad \mathcal{A}(\theta) = \theta \tag{3.6}$$

$$\text{Equisolid:} \quad \mathcal{A}(\theta) = 2\sin\left(\frac{\theta}{2}\right) \quad . \tag{3.7}$$

By inverting the projection model, we find the inclination angle $\theta = \mathcal{A}^{-1}(r_d)$, where $r_d = \sqrt{u_d^2 + v_d^2}$. With the azimuthal angle $\varphi = \arctan(v_d/u_d)$, we calculate the direction $\boldsymbol{d}$ of the 3D ray as

$$\boldsymbol{d} = \begin{pmatrix} \sin(\theta)\cos(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\theta) \end{pmatrix} \quad . \tag{3.8}$$

In section A.1, we derive explicit formulas of the direction $\boldsymbol{d}$ for all four projection models.

Many lenses can be modeled with a single focal point. For wide-angle lenses this is often not the case. The focal point depends on the direction of the ray. A simple but often suitable model is to relate the position of the focal point along the optical axis z to the inclination angle θ:

$$z(\theta) = \left(\frac{1}{\operatorname{sinc}(\theta)} - 1\right) \sum_{i=0}^{N} s_i \theta^{2i} \quad , \tag{3.9}$$

where $N = 2$ is already sufficient for most cases.

3.1.2 LiDAR

LiDAR is short for *light detection and ranging*. It is an active sensor, sending light pulses which are then reflected in the environment and captured again by the sensor. Based on the time between sending and receiving the light pulse, the distance to the reflecting surface can be determined. By combining the distance with the precisely known direction in which the light pulse was sent, the LiDAR provides 3D point measurements. Currently, common LiDARs consist of a few diodes (1-128) that are rotated or their light pulses are deflected by a rotating mirror to measure a wide angular range of the environment. The diodes measure several thousand times per second to create a 3D point cloud.

LiDARs are popular for many robotic applications such as automated driving since they provide precise range information without additional processing. This is a big advantage compared to cameras. Based on the point clouds of a LiDAR, 3D environment models can be created. This information can be used e.g. for localization and mapping ([2, 7]).

3.1.3 Radar

Radar is short for *radio detection and ranging*. The working principle is comparable to LiDAR. It is an active sensor that sends out an electromagnetic signal that is reflected in the environment and captured again by the sensor. The radar cross-section (RCS) is a measure of how much energy is reflected back to the source. It is mainly dependent on the material, shape and size of the reflecting surface. The distance of an object is inferred based on the time the signal needs to return. Further, due to the *Doppler effect*, a shift in the frequency of the signal allows to estimate the relative velocity of the reflecting surface. A major difference between a radar and a LiDAR is the wavelength of the signal. Radars use radio waves which have a longer wavelength than the light waves of LiDARs. This has several consequences: Radar signals are strongly reflected by conductive materials but only weakly by other materials. Automotive radars are less accurate than automotive LiDARs and usually only provide 2D measurements (range and azimuthal angle). An advantage of radars is that objects can be detected at far ranges ($> 200\,\mathrm{m}$) which is not possible with most common LiDARs.

3.1.4 GNSS

Global Navigation Satellite Systems (*GNSS*) such as *GPS* or *GLONASS* consist of multiple satellites that surround the earth in precisely known orbits. They continuously send out signals which can be received on earth. If signals of at least four satellites are received, global positioning and velocity estimation is possible.

3.1.5 Wheel Odometry

Wheel odometry estimates position and orientation of a wheeled platform based on encoder readings from wheels and steering angle readings. By using a precisely known vehicle model, the wheel rotation speeds and the steering angle describe the movement of a vehicle when assuming no slip of the wheels. Wheel odometry is considered to be very reliable and usually provides a high quality movement estimation with low drift. Same as the vehicle model, the wheel odometry is usually defined relative to the center of the rear axle.

3.2 Computer Vision Tools

In this section, we discuss fundamental computer vision methods that are needed for detecting our calibration target in camera images.

3.2.1 Edge Detection

In the field of computer vision, an edge is defined as a sharp change of brightness. Edges are often important because they are indicators of change e.g. change from one to another object, change in depth or change of lighting. Many tasks, such as background segmentation and object detection, focus on these changes. So, edge detection is often used as a data reduction and filtering step. This makes edge detection a fundamental low-level task in image processing.

Edge detection is a well-researched standard task but still considered to be challenging depending on the requirements of the application. The main challenge is to cope with the varying appearance of edges due to changes in lighting, blur and contrast. Further, pixel noise has to be filtered by image smoothing which leads to even more blur on the edges. If an edge is detected, it has to be localized accurately. Usually, the high gradient of an edge stretches over multiple pixels so that determining the exact location is challenging. For many applications, e.g. cartography, even sub-pixel localization accuracy is needed.

In literature, many approaches for edge detection are presented. The detectors can be split into two classes [JS09]. Detectors of the first class use an operator to estimate the first order derivative of an image. Popular first order derivative operators are Prewitt [Pre70], Sobel [DH73] and Robert operator [Dav75]. The positions of maxima and minima of the first order derivative are potential edge points. Detectors of the second class use an estimate of the second order derivative. Edge candidates are the zero crossings of the second order derivative. Finding zero crossings is computationally less expensive than finding local maxima and minima which makes the second class of detectors more efficient [BM12]. The downside of using second order derivatives is the higher influence of noise compared to using first order derivatives.

Finding edge candidates by the previously described operators is only one step for robust and accurate edge detection. Many false positive edges are generated by noise and false negatives occur due to low contrast, to name just a few problems. In the work of Canny [Can86], a multi step process is described to robustly extract relevant edges. This process is named after its inventor and is considered the standard method for edge detection with pixel precision.

Canny Edge Detection

The following details on Canny Edge Detection are taken from [Can86, JS09, BM12] . The process of Canny Edge Detection can be split up into five steps:

1. Smoothing of the image by convolution with a Gaussian kernel.
2. Determination of edge strength and rough orientation by calculating the absolute intensity gradient and its direction.
3. Suppression of edges which have stronger edges in their direct neighborhood.
4. Thresholding to classify points in no ,weak and strong edge.
5. Tracking of edge connectivity to eliminate weak edges.

First Step: Pixel noise can induce high local gradients which are misinterpreted as relevant edges. By smoothing the image, the pixel noise can be reduced. Canny proposed to use a Gaussian kernel which is convolved with the image. On the one hand, the size of the kernel should not be set too large because too much smoothing blurs the edges which deteriorates localization accuracy. On the other hand, the kernel size should be large enough to reduce noise to an acceptable amount.
Second Step: The Canny Edge Detector works on the first order derivative of the image intensity. A vertical and a horizontal Sobel mask are used to estimate the gradients in vertical and horizontal directions, respectively. The absolute gradient at a pixel position is not exactly calculated because the square root which would be needed for the computation of the norm is too expensive for practical applications. Instead, the absolute gradient $|G|$ is approximated by the sum of the absolute vertical $|G_v|$ and horizontal gradient $|G_h|$ ($|G| \approx |G_v| + |G_h|$). The orientation of the gradient is calculated by the 2-argument arctangent $\mathrm{atan2}(G_v, G_h)$. The orientation is reduced to four angle bins which represent vertical, horizontal and two diagonal directions.
Third Step: Ideally, an edge is a step in intensity over one pixel. Locating an ideal edge is trivial. The gradient peaks at the single pixel position of the edge. But, due to blur, an edge appears as a smooth step over multiple pixels. So, the gradient is high over multiple pixels. It is assumed that the true edge location is the position with the highest absolute gradient. To isolate the positions with highest absolute gradients, non-maximum suppression is applied. For each pixel position, the value of the absolute gradient is compared to the values of the two neighboring pixels in edge direction which is orthogonal to the gradient orientation. If one of the two neighboring pixels has a higher absolute gradient, the considered pixel position is suppressed by setting the gradient to zero. This process is also called edge thinning since the edges are reduced to single pixel width.
Fourth Step: Not all pixel positions with a non-zero absolute gradient are edge positions. The absolute value of the gradient defines the saliency of an edge. Two thresholds are used to classify pixel positions based on the absolute gradient value. If the value is smaller than the lower threshold then the pixel position is classified as no edge. A weak edge has a value between the lower and the upper threshold. For a value higher than the upper threshold the pixel position is categorized as strong edge.
Fifth Step: A final consideration to decide on weak edges is needed. On the one hand, a weak edge can be the result of noise, and therefore, is irrele-

vant. On the other hand, a relevant edge can have a moderate gradient due to lighting, contrast and blur. A simple criterion to reason about weak edges is connectivity to a strong edge. Weak edges are preserved if they are connected to a strong edge and discarded if not.

Steger Method

Canny Edge Detection provides reliable edge detections but only with pixel precision. Steger developed a method to detect and localize curvilinear structures with sub-pixel precision [Ste98]. His approach can also be used to detect edges because edges appear as lines in the absolute gradient image. The following details are based on his work [Ste98].

A good starting point for the Steger method is to use the Canny Edge Detector for initial edge candidates. These detections are refined by the following process:
For a candidate edge pixel, the gradient is calculated in the neighborhood by e.g. the Sobel operator. The direction $\boldsymbol{n} = (n_x, n_y)$ orthogonal to the edge is calculated by means of the Hessian $\boldsymbol{H}_{|G|}$ of the absolute gradient $|G|$ at that pixel position:

$$\boldsymbol{H}_{|G|}(x, y) = \begin{pmatrix} |G|_{xx} & |G|_{xy} \\ |G|_{xy} & |G|_{yy} \end{pmatrix} \tag{3.10}$$

The eigenvector with the largest corresponding eigenvalue of the Hessian $\boldsymbol{H}_{|G|}$ points in orthogonal edge direction $\boldsymbol{n}$. The second order derivatives $|G|_{xx}$, $|G|_{xy}$ and $|G|_{yy}$ needed for the Hessian are reliably calculated by using a local bicubic facet model (two-dimensional cubic polynomial fit) for the absolute gradient $|G|$ [HWL83]. The edge is located at the zero-crossing along the orthogonal edge direction. By using a second order Taylor polynomial the sub-pixel offset for the edge position can be derived:

$$\begin{aligned} &(\Delta p_x, \Delta p_y) = (t n_x, t n_y) \ , \\ &\text{with} \quad t = -\frac{G_x n_x + G_y n_y}{G_{xx} n_x^2 + 2G_{xy} n_x n_y + G_{yy} n_y^2} \ . \end{aligned} \tag{3.11}$$

3.2.2 Circle Detection

Detecting circles in images is a standard task in computer vision. There are many applications for circle detection in industry e.g. sorting tasks in assembly lines and automatic hole measuring for quality control.

The most popular method for circle detection in images is the Hough Transform. The Hough Transform is a general method for feature extraction [Hou62, DH71]. It is only practical for features described by few parameters since efficiency of the Hough Transform does not scale well with the complexity of the features. A circle can be parameterized by its center point (x_c, y_c) and its radius R. A point (x, y) on the circle fulfills the circle equation

$$(x - x_c)^2 + (y - y_c)^2 = R^2. \tag{3.12}$$

The standard implementation of the Circle Hough Transform consists of multiple steps [YPIK90]:
First, edge points have to be extracted e.g. by the Canny Edge Detector (see subsubsection 3.2.1). Each edge point might be a point on a circle. Additionally, the normal direction of the edge points are estimated (e.g. by the Hessian as explained for the Steger method in subsubsection 3.2.1). The direction information is used in a later stage to increase efficiency.
As a next step, each edge point is transformed to the circle parameter space. What does that mean? A circle is represented by its three parameters. We define the circle parameter space as a 3D space with the dimensions x_c, y_c and R. So, each 3D point (x_i, y_i, R_i) in the circle parameter space represents a circle with center point (x_i, y_i) and radius R_i. A single edge point does not uniquely represent a circle but limits candidates to circles that intersect the edge point. For a moment, let's assume that the radius of the circles we are searching for is fixed. Then a single edge point transforms to a circle in the parameter space (see Equation 3.12). If the radius is not fixed, it transforms to a circular cone whose axis is perpendicular to its base (right circular cone) [DH71] (see Figure 3.2 bottom left).
The Hough Transform uses a discretized parameter space. Each edge point votes for the cells in the discretized parameter space which intersect the parameter cone. A so-called Hough Accumulator is used to accumulate all votes in the parameter space. At this point, the normal direction of the edge points

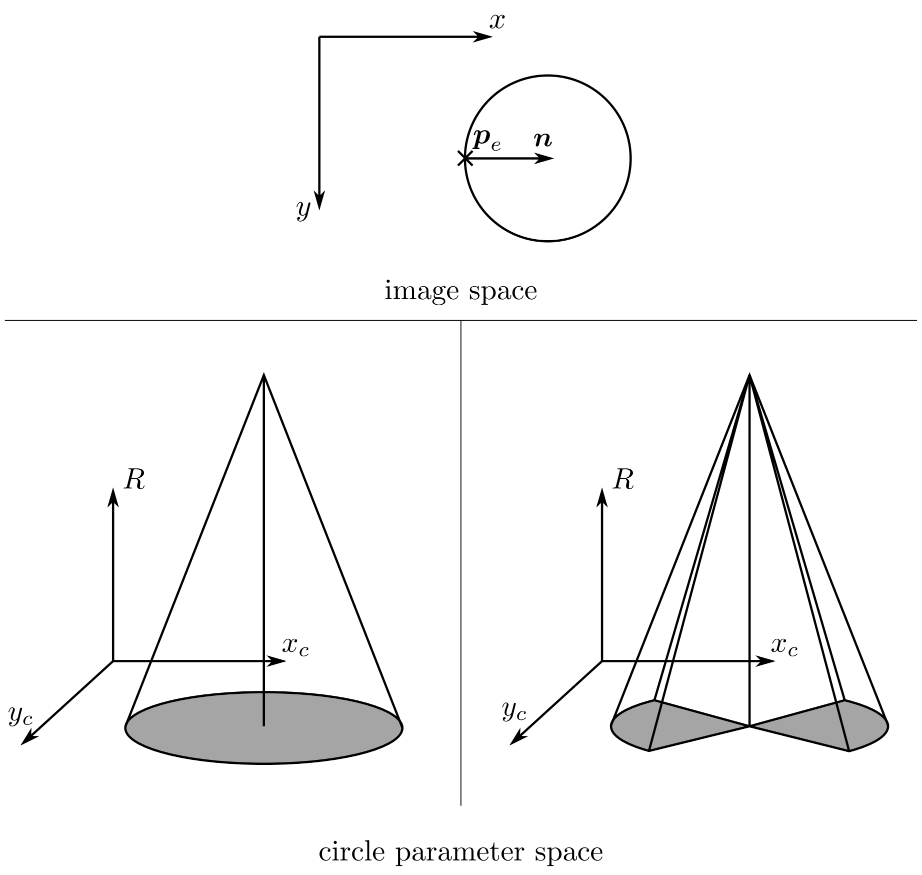


Figure 3.2: 2D image space (top) and 3D circle parameter space (bottom). The edge point $\boldsymbol{p_e}$ with normal direction $\boldsymbol{n}$ is transformed from the image space into the circle parameter space. If no direction information is used, the edge point $\boldsymbol{p_e}$ is represented as a right circular cone with its symmetry axis being parallel to the $\boldsymbol{R}$ axis (bottom left). If the edge direction is additionally taken into account, the transformation leads to the reduced cone (bottom right). In both cases, the gray surfaces only serve for better visualization and are not part of the transformation.

can be used to reduce the cones (see Figure 3.2), and therefore, reduces the number of votes significantly. Cells with high vote counts represent prominent circles in the image.
The final step of the Circle Hough Transform is to find the locations of high vote counts. A central question is, how high is a high vote count? It depends on the expected number of pixels which should lie on a valid circle. Usually, the threshold for accepting a circle is set low to allow for detecting partly occluded circles and to compensate for missed edge points. To prevent detecting circles which are close to local maxima in parameter space, non-maximum suppression is used (as described in subsubsection 3.2.1 for edge detection).

Depending on the preset ranges and resolution of the Hough Accumulator, the standard Circle Hough Transform can be very expensive in memory and computation. Therefore, many variations have been developed which are more efficient than the original method. One of these variations is the Adaptive Hough Transform ([IK87]) which uses an intelligent accumulator that adapts its range and resolution to the underlying data. By only considering the relevant part of the parameter space, the adaptive accumulator takes less memory and the maximum search can be performed significantly faster.

Since the Hough Transform uses a discretized parameter space, the circle parameters are determined only with limited precision. In many tasks, we want to further optimize the circle parameters for improved precision.

3.2.3 ArUco Marker System

The ArUco marker system is a special type of fiducial marker system that originally was developed to estimate camera poses in an environment. To estimate a camera pose, correspondences of points in the environment and their projection in the camera image are needed. This can also be done without markers but a dedicated marker system offers an easy way to increase speed, robustness and precision of the estimate. For that reason, many different fiducial marker systems have been presented in literature (Figure 3.3).

The most popular class of markers is the square-based marker system. Markers of this class are composed by a black square on a white background and an identification code inside the square. The black square makes efficient prefiltering possible which is essential for fast detection of the markers. The code makes each marker unique and usually offers redundancy to compensate for wrong detections.

The ArUco marker system [GJMSMCMJ14] is a square-based marker system with a binary identification code (Figure 3.4). The marker system is special in its configuration of the binary identification code. Usually, a marker systems comes with a fixed dictionary which contains all available markers. This is problematic because if more markers are needed than the dictionary contains, markers have to be used more than once, and so, uniqueness of

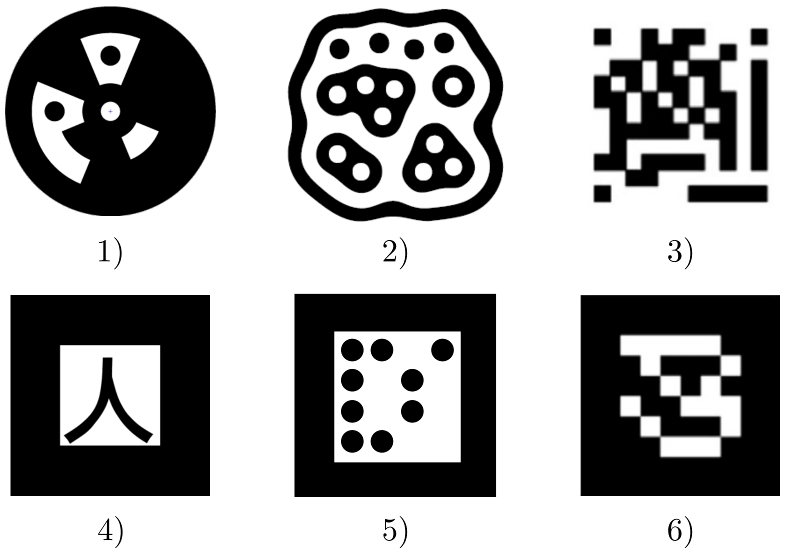


Figure 3.3: Different fiducial markers: 1) InterSense [NF02], 2) ReacTIVision [JGAK07], 3) VisualCode [RG04], 4) ARToolKit [Kat], 5) SCR [ZGN01] and 6) ARTag [Fia09].

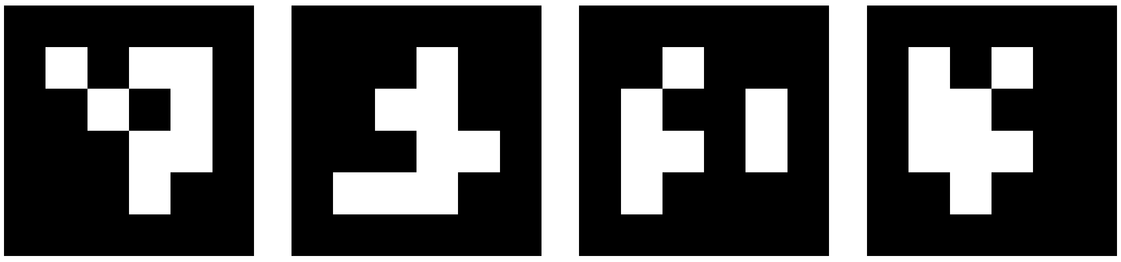


Figure 3.4: Four different ArUco markers with code bit size 4x4.

markers is not given anymore. Further, if less markers are needed than the dictionary contains, the inter-marker distance which is essential for detecting and compensating for wrong detections is suboptimal. The dictionary of the ArUco marker system can be generated for an individual number of markers. This maximizes the inter-marker distance which results in a minimal inter-marker confusion rate.

The authors of [GJMSMCMJ14] propose the following process to generate an individual ArUco dictionary:
The dictionary is generated by using a stochastic algorithm since checking the complete search space is unfeasible even for codes with a low number of bits. This might result in suboptimal results. The generation process is iterative.

We start with an empty dictionary $\mathcal{D}$. A code is randomly generated and only added to the dictionary if the distance to all the codes in the dictionary is higher than a threshold τ. The distance between two markers M_1 and M_2 is defined as the minimum of hamming distances of M_1 and the by $k \cdot 90°$ rotated version of M_2, where $k \in [0, 1, 2, 3]$. Further, the proposed marker is only added to the dictionary if its distance to its rotated versions is also higher than the threshold τ. This is important to robustly estimate the orientation of the marker. The threshold τ is set to the maximal possible inter-marker distance in the beginning and is then slowly decreased to guarantee that the dictionary achieves the aimed size.

In the following, the detection and identification process, as proposed in the original publication [GJMSMCMJ14], is summarized:
A gray-scale image serves as input. As a first processing step, edges are detected e.g. by using the Canny Edge Detector. Next, contours are extracted based on the edge points by using the method presented in [SA85]. The resulting contours are approximated by polygons with the algorithm introduced in [DP73]. Only polygons which are approximated by four vertexes are considered in the further process.
At this point, good marker candidates are efficiently generated without considering the inside of the polygons. To further filter out outliers and extract the code of the markers, the perspective projection of the polygon is calculated by finding the homography matrix [HZ03]. The resulting square is divided into a grid of size $(n+2)\times(n+2)$, where $n \times n$ is the code bit size (see Figure 3.4). Next, a threshold for binarization is found with the method described in [Ots79] which is optimal for the bimodal pixel value distribution of a marker. If the average pixel value in a cell is lower than this threshold, the cell value is set to zero. Else, the cell value is set to one. A marker candidate is rejected if not all of its outer cells are zero. The inner cell values define the binary identification code. If the detected code or one of its rotated versions is in the dictionary $\mathcal{D}$, a valid marker was found. If this is not the case, we try to correct for detection errors. All codes in the dictionary have a distance of at least $\hat{\tau}$ to each other, where $\hat{\tau}$ is the threshold of accepting a new marker in the dictionary at the end of the stochastic algorithm. So, a practical way to correct for detection errors is to associate the incorrect code with the closest code in the dictionary if the distance is equal to or smaller than $\lfloor(\hat{\tau}-1)/2\rfloor$. This correction method results in high reliability for a great variety of scenarios.

3.3 Statistical Tools

The principle of causality states that every real event has a cause. From causality can be inferred that there must be a logical relationship between two events, the cause and the effect. Cause precedes the effect. Assuming causality always holds, everything would be deterministic. In practice, the lack of information and the complexity of the logical relationship between cause and effect motivate the concept of uncertainty [Rub07].

3.3.1 Probability Theory

Probability theory is a mathematical tool to model and work with uncertainties. The following explanations are based on [Was13] and [Bis06].

Probability

We consider an experiment which can have different outcomes. The set of possible outcomes is defined as the *sample space* Ω. *Sample outcomes* or *realizations* are points ω in Ω. *Events* are subsets of Ω.
For example, we perform an experiment with a coin which has two sides, heads (H) and tails (T). The coin is tossed twice. So, four different sample outcomes ω_1 = TT, ω_2 = TH, ω_3 = HT and ω_4 = HH are possible. The sample space is defined by $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$. The event A_T that the first toss is tails can be written as $A_T = \{\omega_1, \omega_2\}$.

The *probability distribution* $\mathbb{P}$ assigns a real number $\mathbb{P}(A)$ to an event A. $\mathbb{P}$ satisfies three axioms:

1. $\mathbb{P}(A) \geq 0$ for all $A \subset \Omega$.
2. $\mathbb{P}(\Omega) = 1$.
3. For disjoint events $A_1, A_2, \ldots$ the following holds:
   $\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i)$.

For the previously mentioned coin experiment, the probability distribution $\mathbb{P}$ assigns for every sample outcome ω_i the same probability $\mathbb{P}(A = \omega_i) = 0.25$, assuming a fair coin. For the event $A_T = \{\omega_1, \omega_2\}$ that the first toss is tails, we calculate $\mathbb{P}(A_T) = \mathbb{P}(A = \omega_1 \cup A = \omega_2) = \mathbb{P}(A = \omega_1) + \mathbb{P}(A = \omega_2) = 0.5$.

An important concept of probability theory is *independence* of two events A and B. Event A is independent of event B if one event's occurrence does not influence the probability that the other event will occur. Formally, independence is defined as follows:
Two events A and B are independent if and only if $\mathbb{P}(AB) = \mathbb{P}(A)\mathbb{P}(B)$.
For example, when tossing a coin twice, the outcome of each toss is independent of the other. Let's define the event A as tails for the first toss and event B as tails for the second toss. Because of independence, we find $\mathbb{P}(AB) = \mathbb{P}(A)\mathbb{P}(B) = 0.5 \cdot 0.5 = 0.25$.

In the case that two events A and B are not independent, the occurrence of one event affects the probability of the other event to occur. The *conditional probability* is an important concept to model this dependency:
The conditional probability of A given B is defined as $\mathbb{P}(A|B) = \frac{\mathbb{P}(AB)}{\mathbb{P}(B)}$, if $\mathbb{P}(B) > 0$.
Again, let's use the coin experiment as an example. We define A as the event that both tosses are tails $A = \{\omega_1\}$ and B as the event that the first toss is tails $B = \{\omega_1, \omega_2\}$. The conditional probability $\mathbb{P}(A|B)$ of both tosses are tails given that the first toss is tails can be computed as $\mathbb{P}(A|B) = \frac{\mathbb{P}(AB)}{\mathbb{P}(B)} = \frac{\mathbb{P}(\{\omega_1\})}{\mathbb{P}(\{\omega_1,\omega_2\})} = \frac{0.25}{0.5} = 0.5$.

For fixed B, $\mathbb{P}(A|B)$ satisfies the previously introduced three axioms, and therefore, is a probability distribution as well. Previously introduced independence of two events A and B can also be defined by means of the conditional probability:
If and only if $\mathbb{P}(A|B) = \mathbb{P}(A)$ holds, A and B are independent.
Note that, in general, $\mathbb{P}(A|B) = \mathbb{P}(B|A)$ does not hold. The relationship between the two conditional probabilities is defined by the *Bayes' Theorem*:
$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$.
$\mathbb{P}(A)$ is called the prior probability, $\mathbb{P}(A|B)$ the posterior probability and $\mathbb{P}(B|A)$ the likelihood. The denominator $\mathbb{P}(B)$ can be interpreted as a normalization constant which assures that the posterior is a valid probability.

Random Variable

Working directly with sample spaces and events is impractical for many problems. Random variables serve as a mathematical tool to model and work with uncertainty in a convenient and powerful way:
The mapping $X\colon \Omega \longrightarrow \mathbb{R}$ that assigns a real number $X(\omega)$ to each sample outcome ω is called *random variable*.
For example, we can define the random variable X to be the number of tails when tossing a coin twice. The mapping would be

$$X(\omega) = \begin{cases} 0, & \text{if } \omega = HH \\ 1, & \text{if } \omega = HT \text{ or } \omega = TH \\ 2, & \text{if } \omega = TT \end{cases} . \tag{3.13}$$

The function $F_X\colon \mathbb{R} \longrightarrow [0, 1]$ defined by $F_X(x) = \mathbb{P}(X \le x)$ is called *cumulative distribution function* (CDF). A CDF has the following three properties:

1. F_X is non-decreasing: $x_1 < x_2 \Rightarrow F_X(x_1) \le F_X(x_2)$.
2. F_X is a right-continuous function:
   $F_X(x) = \lim_{y \to x} F_X(y)$, for all x with $y > x$.
3. F_X is normalized: $\lim_{x \to -\infty} F_X(x) = 0$ and $\lim_{x \to \infty} F_X(x) = 1$.

Figure 3.5 shows the CDF of the random variable X (Equation 3.13) for our coin experiment.

We differentiate between two kinds of random variables:
A *discrete random variable* X maps to countably many values $X\colon \Omega \longrightarrow \{x_1, x_2, ..., x_n\}$. $f_X(x) = \mathbb{P}(X = x)$ is called *probability function* or *probability mass function*.
A *continuous random variable* X maps to infinitely many values. A so called *probability density function* (PDF) f_X must exist with the following properties:

1. $f_X(x) \ge 0$, for all x.
2. $\int_{-\infty}^{\infty} f_X(x)dx = 1$.

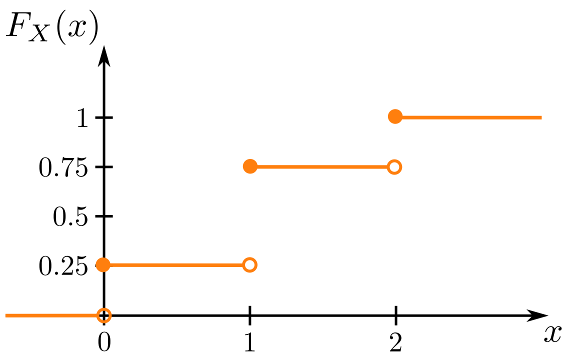


Figure 3.5: CDF of the random variable X which is defined as the number of tails occurring in a coin experiment when tossing twice [Was13].

3. $\mathbb{P}(a < X < b) = \int_a^b f_X(x)dx$, for all $a \leq b$.

An example for a discrete random variable is given by Equation 3.13 for the coin experiment. An important example for a continuous random variable is the so-called *normal* or *Gaussian distribution* which is defined by the PDF

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad , \tag{3.14}$$

with the parameters μ and σ^2 being the mean and variance, respectively (see subsection 3.3.2). Figure 3.6 visualizes three different PDFs for normal distributions with different parameter sets.

As for events and their probability, we can define useful relations between random variables:
Two random variables X and Y are independent if
$\mathbb{P}(X \in A, Y \in B) = \mathbb{P}(X \in A)\mathbb{P}(Y \in B)$, for every set A and B.
If two random variables X and Y are not independent, their relationship can be described by $f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}$, if $f_Y(y) > 0$, where f represents the probability mass function in case of discrete random variables and the probability density function in case of continuous random variables.

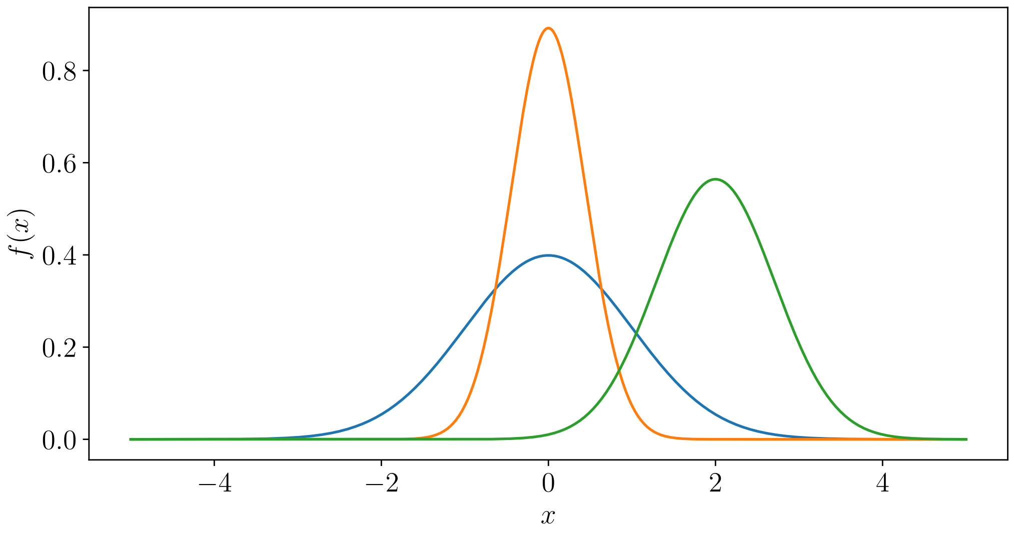


Figure 3.6: PDFs of three normal distributions with different parameters: $\mu_{\text{orange}} = 0$, $\sigma^2_{\text{orange}} = 0.2$, $\mu_{\text{blue}} = 0$, $\sigma^2_{\text{blue}} = 1$, $\mu_{\text{green}} = 2$, $\sigma^2_{\text{green}} = 0.5$.

3.3.2 Mean and Variance

The *mean* and the *variance* are characteristic values of a distribution. The mean $\mathbb{E}[X]$ and the variance $\mathbb{V}(X)$ of a random variable X with probability density function $f(x)$ are defined as follows:

$$\mathbb{E}[X] = \int_{\mathbb{R}} x f(x) dx \tag{3.15}$$

$$\mathbb{V}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \int_{\mathbb{R}} (x - \mathbb{E}[X])^2 f(x) dx \quad . \tag{3.16}$$

The *covariance* $Cov(X, Y)$ of two random variables X and Y is another characteristic value that can be used to measure how strong the linear relation between two random variables is. With the means μ_X and μ_Y, the covariance $Cov(X, Y)$ is defined as

$$Cov(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] \quad . \tag{3.17}$$

For random variables $X_1, X_2, ..., X_n$ we define the *covariance matrix* $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \begin{pmatrix} Cov(X_1, X_1) & \dots & Cov(X_1, X_n) \\ \vdots & \ddots & \vdots \\ Cov(X_n, X_1) & \dots & Cov(X_n, X_n) \end{pmatrix} . \tag{3.18}$$

In practice, the mean, variance and covariance are often helpful to roughly describe a distribution. Usually, when we are given data, we do not know the underlying distribution. So, we cannot use Equation 3.15 - 3.17 to calculate the characteristic values. We can use the following estimates by drawing n samples from the population:
Given n samples $\{x_1, x_2, ..., x_n\}$ from the random variable X, the *sample mean* $\overline{X}$ and the *sample variance* S^2 are defined as

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.19}$$

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{X})^2. \tag{3.20}$$

The sample covariance Q for random variables X and Y is defined as

$$Q = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y}), \tag{3.21}$$

where x_i and y_i are the outcomes of the random variables X and Y for the i-th sample of the underlying population.

3.3.3 Principal Component Analysis

Principal Component Analysis (PCA) is a method for analyzing data and finding underlying patterns. PCA is an important technique used in many applications such as data compression, dimension reduction and feature extraction.

The main idea of PCA is as following:
Let's assume we have observations of different random variables at hand. PCA finds a linear transformation to represent the data by a set of new uncorrelated random variables. We call these uncorrelated random variables *principal components*. The characteristic about the linear transformation is that the variance of each principal component is maximized in an iterative manner: The first principal component has the largest possible variance. The second principal component maximizes the variance as well but under the condition that it is orthogonal to the first principal component. This process is iteratively performed for all principal components.

Mathematically, PCA can be formulated as following [Jol02, Bis06]:
Let $X_1, ..., X_n$ be n random variables. By drawing m times from the n random variables, we collect observation data which is summarized in $\boldsymbol{X}$ where the i-th row represents the i-th drawing and the j-th column denotes the j-th random variable X_j. We assume that the sample mean for each column is zero. Otherwise, we shift the date to make it zero-mean. A linear transformation maps each row vector $\boldsymbol{x}_i$ to the new system consisting of p principal components:

$$\tilde{x}_{i,k} = \boldsymbol{u}_k{}^T \cdot \boldsymbol{x}_i, \quad i = 1, ..., m \text{ and } k = 1, ..., p \ , \tag{3.22}$$

where $\boldsymbol{u}_k{}^T$ is a n dimensional unit row vector ($\|\boldsymbol{u}_k\| = 1$) and $\tilde{x}_{i,k}$ is the k-th coordinate in the new system of the transformed i-th row vector $\boldsymbol{x}_i$. The first principal component shall have maximized variance. Let $\boldsymbol{Q}$ be the sample covariance matrix calculated based on the observations of the random variable $X_1, ..., X_n$. The variance of the first principal component can be calculated by $\boldsymbol{u}_1{}^T \boldsymbol{Q} \boldsymbol{u}_1$. So, $\boldsymbol{u}_1$ can be found by solving the maximization problem

$$\boldsymbol{u}_1 = \underset{\|\boldsymbol{u}_1\|=1}{\arg\max}\, \boldsymbol{u}_1{}^T \boldsymbol{Q} \boldsymbol{u}_1 \quad . \tag{3.23}$$

By using a Lagrange multiplier, it can be shown that $\boldsymbol{u}_1$ is the eigenvector of $\boldsymbol{Q}$ with the highest eigenvalue [Bis06]. Further, one can show that the following principal component can be calculated by determining the eigenvector with the second highest eigenvalue and so on for all principal components.

3.3.4 Error Propagation

Error propagation deals with the question of how random errors of some input signals propagate through a system and how they affect the outputs. The following details are based on the works [Arr98] and [Tel01].

Let $g(\cdot)$ be a function that models a system taking the random variable X as input and transforming it to the random variable $Y = g(X)$ that we call the output. We assume to know the system represented by $g(\cdot)$ and the distribution of the input X. Error propagation aims at finding the distribution of the output Y. Theoretically, this distribution can be calculated (see [Was13]). In practice, even for simple inputs and systems, the calculations quickly become very complex. Therefore, an approximation of the output distribution for any input and non-linear function $g(\cdot)$ is needed.

In the previous subsection 3.3.2, we introduced the mean and variance of a random variable as characteristic values. For the case of a normal distributed random variable, the mean and variance fully define the distribution (see Equation 3.14). Let's assume a linear system $g(x) = a + bx$ with parameters $a \in \mathbb{R}$ and $b \in \mathbb{R}\backslash\{0\}$. For a normal distributed input X, the output $Y = g(X)$ can be shown to be normal distributed as well with mean $\mu_Y = a + b\mu_X$ and variance $\sigma_Y^2 = b^2\sigma_X^2$ [JW02]. In the case that $g(x)$ is a non-linear function, the output will not be normal distributed. When considering $g(x)$ only in a small region around the mean μ_X, we can approximate $g(x)$ by linearization [BHWM06]:

$$g(x) \approx g(\mu_X) + \left.\frac{dg(x)}{dx}\right|_{x=\mu_X} (x - \mu_X) \quad . \tag{3.24}$$

So, the output Y can be approximated in a small range by a normal distribution with mean μ_Y and variance σ_Y^2:

$$\mu_Y = g(\mu_X) \tag{3.25}$$

$$\sigma_Y^2 = \left(\left.\frac{dg(x)}{dx}\right|_{x=\mu_X} \right)^2 \sigma_X^2 \quad . \tag{3.26}$$

Even if X is not normal distributed, we can propagate the mean and the variance of X through $g(\cdot)$.

The remaining question is: when is this method valid? To answer this question, we consider two aspects:
First, we assumed to know the mean μ_X of X. Usually, this is not true. Sometimes, we have data to calculate the sample mean as an estimate for μ_X (see Equation 3.19). Another scenario would be to only have a single measurement and the variance of the underlying distribution of X. Then, we can only use this single measurement and hope that it is close to the mean.
Second, we linearized the function $g(\cdot)$. The linearization should be valid within at least one standard deviation σ_X around the mean.
In practice, usually, the variance σ_X^2 is small so that both, the estimate of μ_X and the linearzation of $g(\cdot)$, are valid. Therefore, propagating the mean and the variance of X through $g(\cdot)$ is often a useful method to describe the distribution of the output.

Up to this point, we only considered a single input X and a single output Y which is a limitation for many real problems. Let $X_1, X_2, \ldots, X_m$ be m random variables which serve as inputs for the system represented by $\boldsymbol{g}\colon \mathbb{R}^m \longrightarrow \mathbb{R}^n$ with n outputs $Y_1, Y_2, \ldots, Y_n$. Then, the mean vector of the outputs $\boldsymbol{\mu_Y} = (\mu_{Y_1}, \mu_{Y_2}, \ldots, \mu_{Y_n})^T$ can be calculated by

$$\boldsymbol{\mu_Y} = \boldsymbol{g}(\boldsymbol{\mu_X}) \ , \tag{3.27}$$

with $\boldsymbol{\mu_X} = (\mu_{X_1}, \mu_{X_2}, \ldots, \mu_{X_m})^T$ being the mean vector of the inputs.
The linearization of $\boldsymbol{g}(\cdot)$ around the mean $\boldsymbol{\mu_X}$ can be written as

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{x}) &= \boldsymbol{g}(\boldsymbol{\mu_X}) + [\nabla \boldsymbol{g}(\boldsymbol{x})]^T \Big|_{\boldsymbol{x}=\boldsymbol{\mu_X}} (\boldsymbol{x} - \boldsymbol{\mu_X}) && (3.28)\\ &= \boldsymbol{g}(\boldsymbol{\mu_X}) + \begin{pmatrix} \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial x_1} & \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial x_2} & \ldots & \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial x_m} \end{pmatrix} \Bigg|_{\boldsymbol{x}=\boldsymbol{\mu_X}} (\boldsymbol{x} - \boldsymbol{\mu_X}) && (3.29)\\ &= \boldsymbol{g}(\boldsymbol{\mu_X}) + \boldsymbol{J}(\boldsymbol{x} - \boldsymbol{\mu_X}) \ , && (3.30) \end{aligned}$$

where $\boldsymbol{J}$ is the *Jacobian matrix* derived at $\boldsymbol{\mu_X}$. Then, the covariance matrix $\boldsymbol{\Sigma_Y}$ of the outputs is calculated by

$$\boldsymbol{\Sigma_Y} = \boldsymbol{J}\boldsymbol{\Sigma_X}\boldsymbol{J}^T \ , \tag{3.31}$$

with the input covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{X}}$.

A frequently used simplification of Equation 3.31 is based on the assumption that both, the input and output variables, are *uncorrelated*. The covariance matrix for uncorrelated variables is a diagonal matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_m^2 \end{pmatrix}, \tag{3.32}$$

with the variances $\sigma_1^2, \sigma_2^2, \dots, \sigma_m^2$ of the variables. The diagonal elements of Equation 3.31 can be written as

$$\sigma_{Y_i}^2 = \sum_{j=1}^{m} \left(\frac{\partial g_i(\boldsymbol{x})}{\partial x_j} \right)^2 \sigma_{X_j}^2 \text{ , for } i = 1, 2, ..., n. \tag{3.33}$$

3.4 3D Geometry

In the following, we present fundamentals of 3D geometry that are used for our calibration methods.

3.4.1 Distance Measures

Distance measures are an essential concept in 3D geometry. Having a measure of how far apart two geometric objects are, allows to make an assessment of the geometric constellation. This is important e.g. if the geometric constellation shall fulfill certain criteria. In the following, we focus on Euclidean distances between three geometric elements: points, lines and rays.

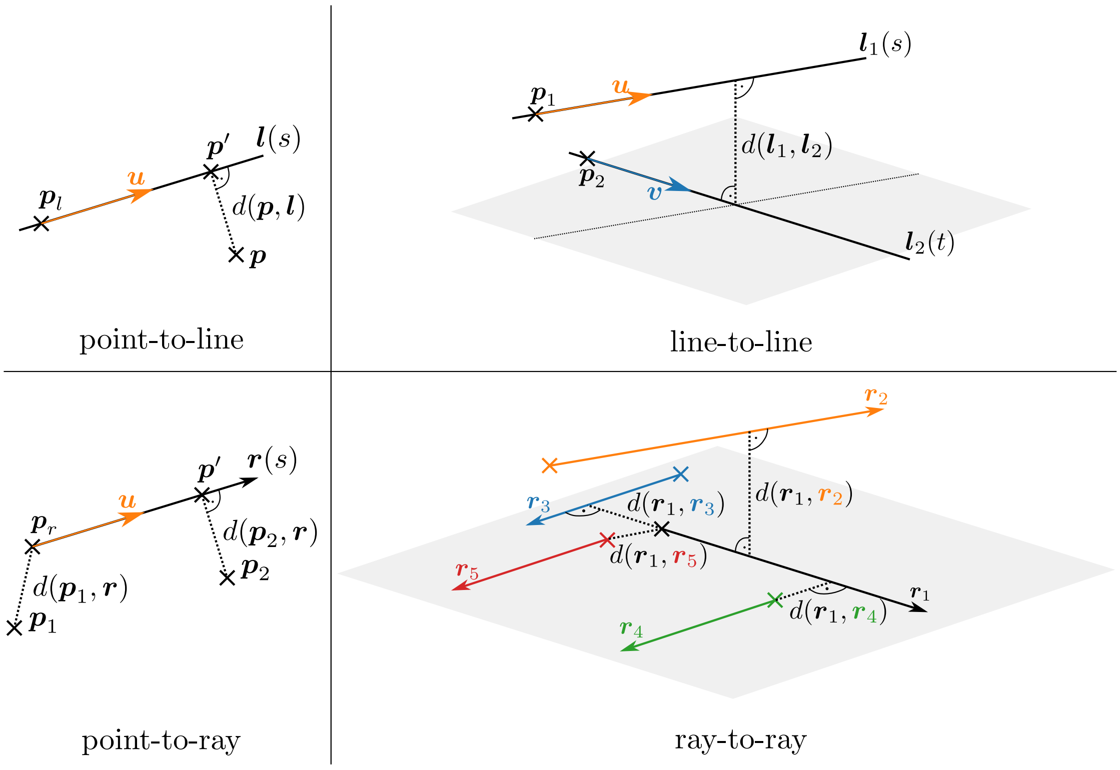


Figure 3.7: Visualization of different distance measures.

Point-to-Point

The distance $d(\boldsymbol{p}_1, \boldsymbol{p}_2)$ between two points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^3$ can be calculated by the Euclidean norm of the connecting vector $\overrightarrow{\boldsymbol{p}_1\boldsymbol{p}_2} = \boldsymbol{p}_2 - \boldsymbol{p}_1$:

$$d(\boldsymbol{p}_1, \boldsymbol{p}_2) = \|\boldsymbol{p}_2 - \boldsymbol{p}_1\| \quad . \tag{3.34}$$

Point-to-Line

Let $\boldsymbol{p} \in \mathbb{R}^3$ be a point and $\boldsymbol{l}(s) = \boldsymbol{p}_l + s\boldsymbol{u}$ be a line with support point $\boldsymbol{p}_l \in \mathbb{R}^3$, direction vector $\boldsymbol{u} \in \mathbb{R}^3$ and control parameter $s \in \mathbb{R}$. Further, let $\boldsymbol{p}'$ be the closest point on the line $\boldsymbol{l}$ to $\boldsymbol{p}$ (see Figure 3.7 top left). We can determine $\boldsymbol{p}'$ by finding the parameter s' for which $\boldsymbol{p}' = \boldsymbol{p}_l + s'\boldsymbol{u}$ holds. The parameter s' can be calculated by projecting the vector $\overrightarrow{\boldsymbol{p}_l\boldsymbol{p}}$ onto the normed direction vector

$\boldsymbol{u}/\|\boldsymbol{u}\|$. Mathematically, projection can be expressed by an inner product. The length of the projection has to be equal to the distance between $\boldsymbol{p_l}$ and $\boldsymbol{p}'$:

$$d(\boldsymbol{p_l}, \boldsymbol{p}') = s'\|\boldsymbol{u}\| = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \cdot (\boldsymbol{p} - \boldsymbol{p_l}) \tag{3.35}$$

$$\Rightarrow s' = \frac{\boldsymbol{u} \cdot (\boldsymbol{p} - \boldsymbol{p_l})}{\|\boldsymbol{u}\|^2} \quad . \tag{3.36}$$

With known s' we can calculate the closest point on the line by

$$\boldsymbol{p}' = \boldsymbol{l}(s') = \boldsymbol{p_l} + s'\boldsymbol{u} = \boldsymbol{p_l} + \frac{\boldsymbol{u} \cdot (\boldsymbol{p} - \boldsymbol{p_l})}{\|\boldsymbol{u}\|^2}\boldsymbol{u} \quad . \tag{3.37}$$

The point-to-line distance $d(\boldsymbol{p}, \boldsymbol{l})$ can be calculated as the point-to-point distance between $\boldsymbol{p}$ and $\boldsymbol{p}'$:

$$d(\boldsymbol{p}, \boldsymbol{l}) = d(\boldsymbol{p}, \boldsymbol{p}') \tag{3.38}$$

$$= \left\| \boldsymbol{p} - \left(\boldsymbol{p_l} + \frac{\boldsymbol{u} \cdot (\boldsymbol{p} - \boldsymbol{p_l})}{\|\boldsymbol{u}\|^2}\boldsymbol{u} \right) \right\| \quad . \tag{3.39}$$

A more compact formula is:

$$d(\boldsymbol{p}, \boldsymbol{l}) = \frac{\|\boldsymbol{u} \times (\boldsymbol{p_l} - \boldsymbol{p})\|}{\|\boldsymbol{u}\|} \quad . \tag{3.40}$$

Line-to-Line

Let $\boldsymbol{l}_1(s) = \boldsymbol{p}_1 + s\boldsymbol{u}$ and $\boldsymbol{l}_2(t) = \boldsymbol{p}_2 + t\boldsymbol{v}$ be two lines with support points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^3$, direction vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^3$ and control parameters $s, t \in \mathbb{R}$ (see Figure 3.7 top right). To find the distance $d(\boldsymbol{l}_1, \boldsymbol{l}_2)$ between the two lines, we first determine the direction of the distance vector. The direction of the distance vector is defined by orthogonality to both line directions $\boldsymbol{u}$ and $\boldsymbol{v}$. Mathematically, we find a vector that is orthogonal to $\boldsymbol{u}$ and $\boldsymbol{v}$ by using the cross product $\boldsymbol{u} \times \boldsymbol{v}$. The projection of any vector connecting the two lines (e.g. $\boldsymbol{p}_2 - \boldsymbol{p}_1$) on this orthogonal vector has a length that equals the distance of the two lines:

$$d(\boldsymbol{l}_1, \boldsymbol{l}_2) = \frac{\|(\boldsymbol{p}_2 - \boldsymbol{p}_1) \cdot (\boldsymbol{u} \times \boldsymbol{v})\|}{\|\boldsymbol{u} \times \boldsymbol{v}\|} \quad . \tag{3.41}$$

Point-to-Ray

Let $\boldsymbol{r}(s) = \boldsymbol{p_r} + s\boldsymbol{u}$ be a ray with origin $\boldsymbol{p_r} \in \mathbb{R}^3$, direction $\boldsymbol{u} \in \mathbb{R}^3$ and control parameter $s \in \mathbb{R}^+$. Two cases have to be considered for calculating the distance $d(\boldsymbol{r}, \boldsymbol{p})$ of the ray $\boldsymbol{r}$ to a point $\boldsymbol{p} \in \mathbb{R}^3$ (see Figure 3.7 bottom left).
In the first case, the closest point on the ray is its origin $\boldsymbol{p}_r$. So, the distance $d(\boldsymbol{p}_1, \boldsymbol{r})$ is equal to the point-to-point distance $d(\boldsymbol{p}_1, \boldsymbol{p}_r)$.
In the second case, the closest point on the ray is not the origin. We can calculate the distance $d(\boldsymbol{p}_2, \boldsymbol{r})$ as the point-to-line distance $d(\boldsymbol{p_2}, \boldsymbol{l})$ with $\boldsymbol{l}(t) = \boldsymbol{p_r} + t\boldsymbol{u}$ where $t \in \mathbb{R}$.
Mathematically, we can use the concept of Equation 3.36 to find out which case applies: For the point-to-line distance, we calculated the control parameter s' which defines the closest point on a line. For a line, the control parameter can be in $\mathbb{R}$. For a ray, the control parameter is constraint to $\mathbb{R}^+$. So, if Equation 3.36 results in a negative control parameter for the ray, we have to switch to point-to-point distance.
We can summarize this in one formula for the distance of a point $\boldsymbol{p}$ to a ray $\boldsymbol{r}$:

$$d(\boldsymbol{p}, \boldsymbol{r}) = \begin{cases} \|\boldsymbol{p_r} - \boldsymbol{p}\| & \text{if} \quad \boldsymbol{u} \cdot (\boldsymbol{p} - \boldsymbol{p_r}) < 0 \\ \frac{\|\boldsymbol{u} \times (\boldsymbol{p_r} - \boldsymbol{p})\|}{\|\boldsymbol{u}\|} & \text{else} \end{cases} \quad . \tag{3.42}$$

Ray-to-Ray

Let $\boldsymbol{r}_i(s) = \boldsymbol{p}_i + s\boldsymbol{u}_i$ with $i = 1, \ldots, 5$ be five rays as depicted in Figure 3.7 at the bottom right. There are four different cases which have to be considered when calculating the distance between two rays.
In the first case ($\boldsymbol{r}_1$ and $\boldsymbol{r}_2$), both closest points on the rays are not their origins. Therefore, the distance between $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ equals the distance of the corresponding lines $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$.
In the second case ($\boldsymbol{r}_1$ and $\boldsymbol{r}_3$), the closest point of $\boldsymbol{r}_1$ is its origin and the

closest point of $\boldsymbol{r}_3$ is not its origin. So, the distance between $\boldsymbol{r}_1$ and $\boldsymbol{r}_3$ can be calculated as the distance between the point $\boldsymbol{p}_1$ and the line corresponding to $\boldsymbol{r}_3$.
The third case ($\boldsymbol{r}_1$ and $\boldsymbol{r}_4$) is similar to the second case but this time the origin of $\boldsymbol{r}_4$ and the corresponding line of $\boldsymbol{r}_1$ is used.
In the fourth and last case ($\boldsymbol{r}_1$ and $\boldsymbol{r}_5$), the closest points are the origins of the rays. So, the distance between the two rays $\boldsymbol{r}_1$ and $\boldsymbol{r}_5$ is the point-to-point distance between their origins $\boldsymbol{p}_1$ and $\boldsymbol{p}_5$.
Let's say, we want to calculate the distance $d(\boldsymbol{r}_a, \boldsymbol{r}_b)$ between any two rays $\boldsymbol{r}_a(s_a) = \boldsymbol{p}_a + s_a\boldsymbol{u}_a$ and $\boldsymbol{r}_b(s_b) = \boldsymbol{p}_b + s_b\boldsymbol{u}_b$. To check which of the four cases applies, we define $s'_a = \boldsymbol{u}_a \cdot (\boldsymbol{p}_b - \boldsymbol{p}_a)$ and $s'_b = \boldsymbol{u}_b \cdot (\boldsymbol{p}_a - \boldsymbol{p}_b)$ (we are only interested in the sign of Equation 3.36). Using s'_a and s'_b we can calculate the distance between $\boldsymbol{r}_a$ and $\boldsymbol{r}_b$ as following:

$$d(\boldsymbol{r}_a, \boldsymbol{r}_b) = \begin{cases} d(\boldsymbol{p}_a, \boldsymbol{p}_b) & = \|\boldsymbol{p}_b - \boldsymbol{p}_a\| & \text{if} \quad s'_a \leq 0, s'_b \leq 0 \\ d(\boldsymbol{l}_a, \boldsymbol{p}_b) & = \frac{\|\boldsymbol{u}_a \times (\boldsymbol{p}_a - \boldsymbol{p}_b)\|}{\|\boldsymbol{u}_{\boldsymbol{a}}\|} & \text{if} \quad s'_a > 0, s'_b \leq 0 \\ d(\boldsymbol{p}_a, \boldsymbol{l}_b) & = \frac{\|\boldsymbol{u}_b \times (\boldsymbol{p}_b - \boldsymbol{p}_a)\|}{\|\boldsymbol{u}_b\|} & \text{if} \quad s'_a \leq 0, s'_b > 0 \\ d(\boldsymbol{l}_a, \boldsymbol{l}_b) & = \frac{\|(\boldsymbol{p}_b - \boldsymbol{p}_a) \cdot (\boldsymbol{u}_a \times \boldsymbol{u}_b)\|}{\|\boldsymbol{u}_a \times \boldsymbol{u}_b\|} & \text{if} \quad s'_a > 0, s'_b > 0 \end{cases}. \tag{3.43}$$

3.4.2 Splines

Splines are a common tool for modeling curves and surfaces. They are used in many fields for e.g. interpolation, approximation and visualization.
A famous application is the design of complex shapes in computer graphics. In this context, splines turn out to be particularly easy to construct for humans and enable fast creation of complex designs. This is due to splines being piecewise polynomial curves. Compared to a single polynomial, a spline often needs a significantly lower polynomial degree for each segment to model complex shapes. This is beneficial for efficient processing and increases numerical stability [PT97]. Further, the piecewise definition of splines makes them well-suited for interactive shape design because selective tuning of parameters can introduce local changes and does not affect the complete curve as in case of a single polynomial.

Mathematically, splines are curves that are piecewise defined by polynomial functions and are differentiable to a preset order. A spline $\boldsymbol{s}(u)$ of degree n (or

order $n + 1$) is defined as following [PBP13]:
Let $a_0, ..., a_m$ be $m + 1$ knots with $a_i \leq a_{i+1}$ and $a_i < a_{i+n+1}$. We call a knot r-fold if it coincides with r knots (a_{i+1} is r-fold if $a_i < a_{i+1} = ... = a_{i+r} < a_{i+r+1}$). In each interval $[a_i, a_{i+1}]$, $\boldsymbol{s}(u)$ is defined by a polynomial of degree less or equal to n. Further, at each r-fold knot the spline $\boldsymbol{s}(u)$ is $n - r$ times differentiable.

A practical representation of a spline $\boldsymbol{s}(u)$ is given by using so-called *basis spline functions* (*B-splines*) N_i^n:

$$\boldsymbol{s}(u) = \sum_i \boldsymbol{c}_i N_i^n(u) \ , \tag{3.44}$$

where $\boldsymbol{c}_i$ is called a *control point*.
A B-spline can be defined by the recurrence formula (often referred to as de Boor or Cox-de Boor algorithm [Cox72, DB72, DB78]) starting with

$$N_i^0(u) = \begin{cases} 1 & \text{if } u \in [a_i, a_{i+1}) \\ 0 & \text{else} \end{cases} \tag{3.45}$$

and succeeding with

$$N_i^n(u) = \alpha_i^{n-1} N_i^{n-1}(u) + (1 - \alpha_{i+1}^{n-1}) N_{i+1}^{n-1}(u) \ , \tag{3.46}$$

where

$$\alpha_i^{n-1} = \frac{u - a_i}{a_{i+n} - a_i} \ . \tag{3.47}$$

For knots a_i with $a_i = a_{i+r}$ the following convention is used:

$$N_i^{r-1} = \frac{N_i^{r-1}}{a_{i+r} - a_i} = \frac{0}{0} = 0 \ . \tag{3.48}$$

Figure 3.8 shows examples for the construction of B-splines. With the recurrence formula, one can show that a B-spline $N_i^n(u)$ is piecewise polynomial of degree n, is positive in (a_i, a_{i+n+1}) and vanishes outside $[a_i, a_{i+n+1}]$.
B-splines show many more useful properties. In the following, a selection of important properties is given [PBP13]:

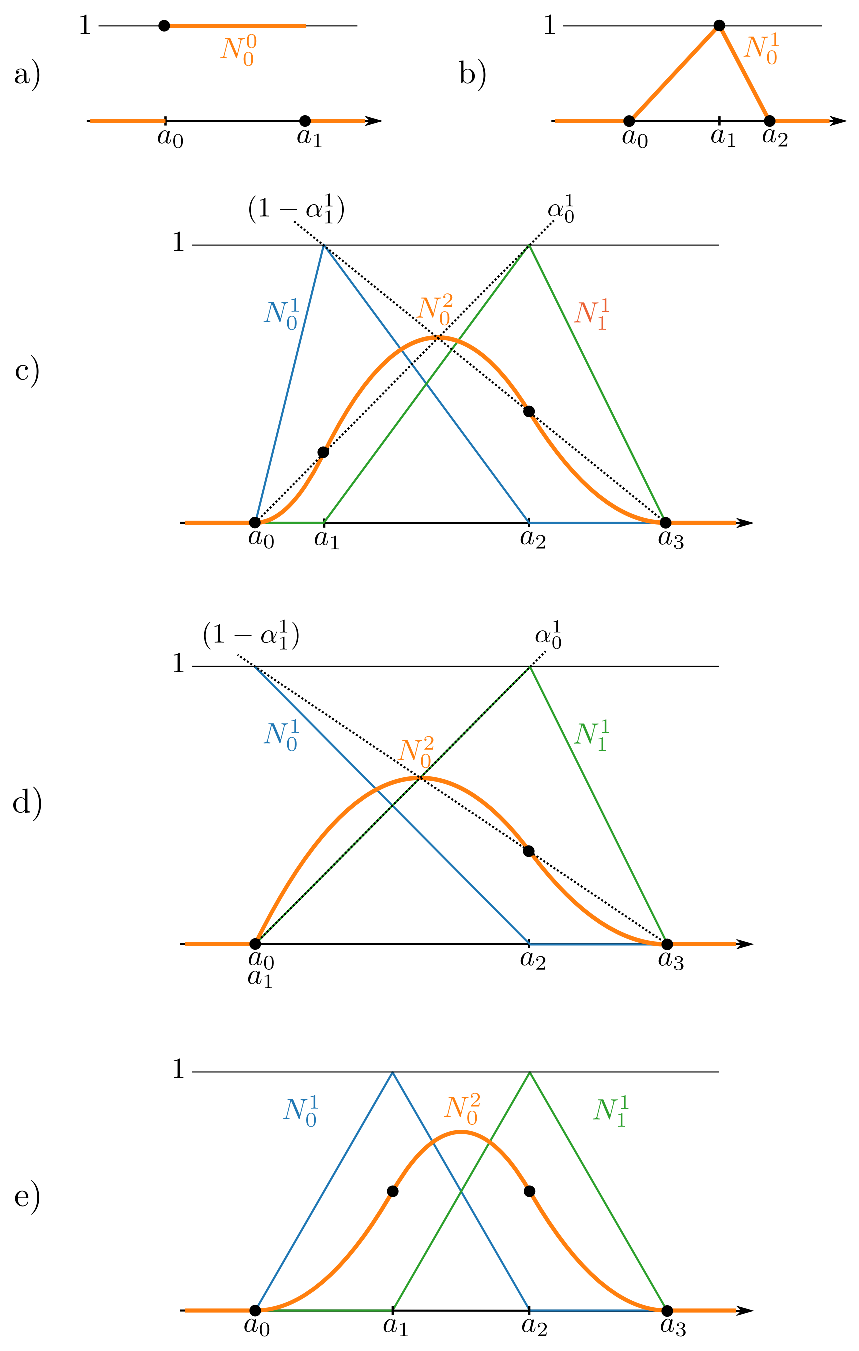


Figure 3.8: Visualization of different B-splines:
a) B-spline of degree 0.
b) B-spline of degree 1.
c) B-splines N_0^1 and N_1^1 of degree 1 and line functions α_0^1 and $(1-\alpha_1^1)$ that are used to construct the B-spline N_0^2 of degree 2 by using the recursive formula of Equation 3.45-3.48.
d) Same as c) but with 2-fold knot $a_0 = a_1$.
e) Uniform B-spline which can be efficiently constructed by convolution and shifting (see Equation 3.51 - 3.53).

- With the knot sequence $a_0, \ldots, a_{m+n+1}$, the according B-splines $N_0^n, \ldots, N_m^n$ of degree n form a basis for all splines of degree n within the interval $[a_n, a_{m+1})$.
- B-splines form a partition of unity,

$$\sum_{i=0}^{m} N_i^n(u) = 1, \text{ for } u \in [a_n, a_{m+1}) \quad . \tag{3.49}$$

- N_0^n and N_m^n are not influenced by the end knots a_0 and a_{m+n+1} over the interval $[a_n, a_{m+1}]$.
- The end points and the end tangents of a spline on the interval $[a_n, a_{m+1}]$ with degree n are the same as its control polygon, if the end knots are duplicate n-times

$$a_0 = a_1 = \ldots = a_n \quad \text{and} \quad a_{m+1} = \ldots = a_{m+n} = a_{m+n+1} \quad .$$

- Any n-th degree spline segment $\boldsymbol{s}_j$ on the interval $[a_j, a_{j+1})$ lies in the convex hull of its $n+1$ control points $\boldsymbol{c}_{j-n}, \ldots, \boldsymbol{c}_j$.
- The derivative of a B-spline of degree n can be calculated by using B-splines of degree $n-1$:

$$\frac{d}{du} N_i^n(u) = \frac{n}{a_{i+n} - a_i} N_i^{n-1}(u) - \frac{n}{a_{i+n+1} - a_{i+1}} N_{i+1}^{n-1}(u). \tag{3.50}$$

If the knots are equally spaced (*uniform* knot sequence) then the corresponding B-splines are called *uniform*. For a knot vector $(0, 1, \ldots, n+1)$ the n-th degree B-spline N^n can be calculated by the recursion formula

$$N^0(u) = \begin{cases} 1 & \text{if } u \in [0,1) \\ 0 & \text{else} \end{cases} \quad , \tag{3.51}$$

$$N^j(u) = N^{j-1} * N^0 \quad , \tag{3.52}$$

where the operator $*$ denotes convolution. Uniform B-splines of the same degree can be calculated very efficiently by shifting [DB78, PBP13]:

$$N_i^n(u) = N^n(u-i) \quad . \tag{3.53}$$

3.4.3 Poses

The concept of poses in the Euclidean 3D space is essential for many fields, such as robotics and 3D vision. A pose holds information about the position and orientation relative to a reference frame. So, a pose can be used to e.g. describe the relative position and orientation of a drone to a landing spot or of two limbs of a robot arm to each other.

Mathematically, we define a pose as a transformation of the *special Euclidean group* $\boldsymbol{SE}(3)$ [Bla10]:
We define a transformation on $\mathbb{R}^3$ as a function $\boldsymbol{f}: \mathbb{R}^3 \to \mathbb{R}^3$.
First, let's look at a transformation represented by a 3×3 matrix $\boldsymbol{R}$ such that a point $\boldsymbol{x}_1 \in \mathbb{R}^3$ is transformed to $\boldsymbol{x}_2 \in \mathbb{R}^3$ by $\boldsymbol{x}_2 = \boldsymbol{R}\boldsymbol{x}_1$. The *general linear group* $\boldsymbol{GL}(3, \mathbb{R})$ is formed by the set of all invertible 3×3 matrices. A subgroup of $\boldsymbol{GL}(3, \mathbb{R})$ is the *orthogonal group* $\boldsymbol{O}(3) \subset \boldsymbol{GL}(3, \mathbb{R})$ which is formed by the set of all orthogonal matrices with determinant ± 1. Transformations with these matrices are *isometries*. An isometry is a transformation that preserves the distance of any point pair. The set of orthogonal matrices with determinant $+1$ forms the *special orthogonal group* $\boldsymbol{SO}(3)$. A matrix $\boldsymbol{R} \in \boldsymbol{SO}(3)$ represents a rotation.
To combine rotation and translation, we increase the dimensions of our transformation matrix by one. We define the 4×4 matrix $\boldsymbol{T}$ as

$$\boldsymbol{T} = \left(\begin{array}{ccc|c} & & & t_x \\ & \boldsymbol{R} & & t_y \\ & & & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \quad , \tag{3.54}$$

with $\boldsymbol{R} \in \boldsymbol{SO}(3)$ and $t_x, t_y, t_z \in \mathbb{R}$. We transform a point $\boldsymbol{x}_1 \in \mathbb{R}^3$ to a point $\boldsymbol{x}_2 \in \mathbb{R}^3$ by using homogeneous coordinates for the points (extend with 1 in additional dimension):

$$\begin{pmatrix} \boldsymbol{x}_2 \\ 1 \end{pmatrix} = \boldsymbol{T} \begin{pmatrix} \boldsymbol{x}_1 \\ 1 \end{pmatrix} \quad . \tag{3.55}$$

The set of all possible transformation matrices $\boldsymbol{T}$ forms the *special Euclidean space* $\boldsymbol{SE}(3)$. Transformations in $\boldsymbol{SE}(3)$ have six degrees of freedom (DoF), three for t_x, t_y, t_z and three for $\boldsymbol{R}$ (columns are orthonormal). $\boldsymbol{SE}(3)$ transformations can be interpreted as defining a position $\boldsymbol{t} = [t_x, t_y, t_z]^T$ and an orientation $\boldsymbol{R}$ relative to a reference coordinate system in $\mathbb{R}^3$. We define poses as transformations in $\boldsymbol{SE}(3)$.

Notation

As stated before, poses are always defined relative to a reference coordinate system $\mathcal{F}_{\text{ref}}$. A pose defines another coordinate system $\mathcal{F}_{\text{A}}$. Often, multiple coordinate systems are in use and we need to clarify for each pose which is the reference coordinate system and what is the denotation of the coordinate system defined by each pose. We use the notation $\boldsymbol{T}_{\mathcal{F}_{\text{ref}},\mathcal{F}_{\text{A}}}$ to define the pose of the coordinate system $\mathcal{F}_{\text{A}}$ in the reference coordinate system $\mathcal{F}_{\text{ref}}$. E.g. let's assume we have two sensors, a LiDAR and a camera. We define two different coordinate systems $\mathcal{F}_{\text{lid}}$ and $\mathcal{F}_{\text{cam}}$. Then, the pose of the camera in the LiDAR coordinate system is denoted as $\boldsymbol{T}_{\mathcal{F}_{\text{lid}},\mathcal{F}_{\text{cam}}}$ and the pose of the LiDAR in the camera frame is given by $\boldsymbol{T}_{\mathcal{F}_{\text{cam}},\mathcal{F}_{\text{lid}}}$. The notation is motivated by the convenient ordering of the subscripts when transforming a point $\boldsymbol{p}_{\mathcal{F}_{\text{A}}}$ defined in coordinate system $\mathcal{F}_{\text{A}}$ to another coordinate system $\mathcal{F}_{\text{B}}$:

$$\begin{pmatrix} \boldsymbol{p}_{\mathcal{F}_{\text{B}}} \\ 1 \end{pmatrix} = \boldsymbol{T}_{\mathcal{F}_{\text{B}},\mathcal{F}_{\text{A}}} \begin{pmatrix} \boldsymbol{p}_{\mathcal{F}_{\text{A}}} \\ 1 \end{pmatrix} \quad . \tag{3.56}$$

The subscripts are ordered to read a chain of transformations $\ldots\boldsymbol{T}_{\mathcal{F}_{\text{D}},\mathcal{F}_{\text{C}}}\boldsymbol{T}_{\mathcal{F}_{\text{C}},\mathcal{F}_{\text{B}}}\boldsymbol{T}_{\mathcal{F}_{\text{B}},\mathcal{F}_{\text{A}}}$ from right to left.
Often, transformations are visualized symbolically by arrows. We use the convention as in Figure 3.9 to define the direction of the arrows.

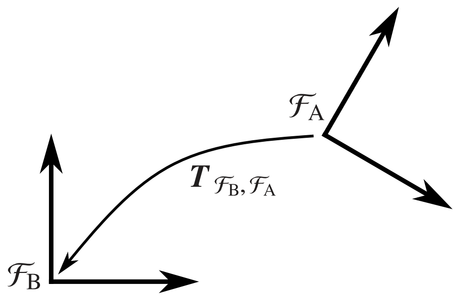


Figure 3.9: Convention to visualize symbolically the transformation $\boldsymbol{T}_{\mathcal{F}_B,\mathcal{F}_A}$.

We summarize the conventions as follows:

$$\boldsymbol{T}_{\mathcal{F}_B,\mathcal{F}_A} \begin{cases} \text{transforms a point from } \mathcal{F}_A \text{ to } \mathcal{F}_B. \\ \text{is the pose that defines } \mathcal{F}_A \text{ in/relative to } \mathcal{F}_B. \\ \text{is visualized symbolically as an arrow from } \mathcal{F}_A \text{ to } \mathcal{F}_B. \end{cases} \tag{3.57}$$

Parameterization

Besides the representation as a 4×4 matrix (Equation 3.54), there are other common parameterizations of poses. In the following, we discuss four popular representations which are used for many applications.

The first parameterization is the one we already introduced in Equation 3.54. Translation and rotation are combined in one 4×4 matrix $\boldsymbol{T}$.
A great advantage of this parameterization is that applying the transformation is simple. A transformation defined by $\boldsymbol{T}$, can simply be performed by matrix multiplication with the extended homogeneous point $[\boldsymbol{p}^T 1]^T$ (see Equation 3.55). Further, combining multiple transformations $\boldsymbol{T}_1, \boldsymbol{T}_2, ..., \boldsymbol{T}_n$ is also given by straightforward matrix multiplication $\boldsymbol{T} = \boldsymbol{T}_n ... \boldsymbol{T}_2 \boldsymbol{T}_1$ (order is important).
The downside of using a 4×4 matrix is that it is not compact with its 16 entries for only six degrees of freedom. Additionally, human readability is not given because of the rotation represented as a 3×3 matrix.

Another parameterization uses *Euler angles* to represent rotations. Euler angles use three values which define three consecutive rotations about three

axes. There are different conventions for the axes. We use the convention yaw φ, pitch θ, roll ψ which is commonly used in robotics. With this convention, a rotation is split in, first, a rotation about the z-axis (yaw), second, a rotation about the modified y-axis (pitch) and, third, a rotation about the modified x-axis (roll).
An advantage of this parameterization is its compactness. Three parameters is the minimum number possible to represent three degrees of freedom. Further, Euler angles are easy to interpret and visualize for humans.
The most important downside of Euler angles is the so-called *gimbal lock*. If two of the three rotation axes are aligned, the rotation is locked in a degenerate two dimensional space. With the yaw, pitch, roll convention, gimbal lock occurs for pitch $\theta = \pm 90°$. In this configuration, roll and yaw are the same. As a result, there is no unique combination of Euler angles to express a 3D rotation in gimbal lock configuration.

The third parameterization we discuss uses the *angle-axis* representation for the rotational part of the transformation. Euler's rotation theorem states, that every rotation or combination of rotations can be expressed by a single rotation around one fixed axis. So, the angle-axis representation defines an axis $\boldsymbol{n}$ and an angle of rotation θ about this axis. A compact representation is to combine the angle and the axis to a 3D vector:

$$\boldsymbol{\theta} = \theta \boldsymbol{n} \quad , \tag{3.58}$$

where $\boldsymbol{n}$ is normalized. So, the rotation angle is encoded in the norm $\|\boldsymbol{\theta}\|$.
As with Euler angles, the parameterization of the angle-axis representation uses the minimal number of parameters to represent the three degrees of freedom. Another advantage is that the representation is well interpretable for humans.
The angle-axis representation is not suited for directly applying rotations. There is a formula, called Rodrigues' rotation formula, which allows for rotating a vector directly but it is not as efficient as working e.g. with rotation matrices. Also, combining multiple rotations in angle-axis representation is not trivial.

As the last parameterization, we discuss representing rotations with *unit* *quaternions*. A quaternion $\boldsymbol{q}$ can be written as a complex number with three imaginary parts i, j, k [Vin11]:

$$\boldsymbol{q} = s + xi + yj + zk \quad \text{, with } s, x, y, z \in \mathbb{R} \quad . \tag{3.59}$$

The relation between the imaginary parts is given by

$$\begin{aligned} ii = jj = kk = ijk &= -1 \quad , \\ ij = \ \ k, \quad jk = \ \ i, \quad ki &= \ \ j \quad , \\ ji = -k, \quad kj = -i, \quad ik &= -j \quad . \end{aligned} \tag{3.60}$$

A more compact and practical representation is to write the imaginary part as a vector so that a quaternion can be written as a pair of a scalar s and a vector $\boldsymbol{v} = [x, y, z]$:

$$\boldsymbol{q} = [s, \boldsymbol{v}] \quad . \tag{3.61}$$

With standard multiplication of complex numbers and the rules given by Equation 3.60, we can derive a short expression for the product

$$\boldsymbol{q}_a \boldsymbol{q}_b = [s_a, \boldsymbol{a}][s_b, \boldsymbol{b}] \tag{3.62}$$
$$= [s_a s_b - \boldsymbol{a} \cdot \boldsymbol{b}, s_a \boldsymbol{b} + s_b \boldsymbol{a} + \boldsymbol{a} \times \boldsymbol{b}] \quad . \tag{3.63}$$

A quaternion $\boldsymbol{q}$ is a unit quaternion if the norm $|\boldsymbol{q}|$ equals to one: $|\boldsymbol{q}| = \sqrt{s^2 + x^2 + y^2 + z^2} = 1$. The inverse of a unit quaternion is equal to its conjugate: $\boldsymbol{q}^{-1} = \overline{\boldsymbol{q}} = s - xi - yj - zk = [s, -\boldsymbol{v}]$.
Unit quaternions can be used to represent rotations. A unit quaternion $\boldsymbol{q}$ which is constructed as

$$\boldsymbol{q} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) n_x i + \sin\left(\frac{\theta}{2}\right) n_y j + \sin\left(\frac{\theta}{2}\right) n_z k \tag{3.64}$$
$$= \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) \boldsymbol{n}\right] \tag{3.65}$$

can be interpreted as a rotation of θ about the axis $\boldsymbol{n} = [n_x, n_y, n_z]$. Let a vector $\boldsymbol{v} \in \mathbb{R}^3$ be encoded in the imaginary part of the quaternion $\boldsymbol{p} = [0, \boldsymbol{v}]$. Then, the imaginary part of the quaternion $\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1} = [0, \boldsymbol{w}]$ holds the vector $\boldsymbol{w}$ that is $\boldsymbol{v}$ rotated by $\boldsymbol{q}$. The combination of two consecutive rotations $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$

can simply be calculated by $\boldsymbol{q} = \boldsymbol{q}_2\boldsymbol{q}_1$.
Quaternions show many advantages over other parameterizations for rotations: One reason why they are used in many applications is that they have no singularities or discontinuities. They are more compact than matrices and allow for very efficient computation of rotations. Additionally, they are superior in numerical stability compared to matrices.
The downside of expressing rotations with unit quaternions is that they are not convenient to interpret for humans.

Interpolation

Interpolation of poses is important for many applications in robotics. E.g. the end effector of a robotic arm shall be moved from a pose $\boldsymbol{T}_0$ to a pose $\boldsymbol{T}_1$. For creating the path between the start and end pose, interpolation is needed.

Interpolation of a pose is usually split up into interpolation of position and interpolation of orientation. In the simplest case, we are given two poses $\boldsymbol{P}_1 = \boldsymbol{T}_{\mathcal{F}_0,\mathcal{F}_1}$ at time t_1 and $\boldsymbol{P}_2 = \boldsymbol{T}_{\mathcal{F}_0,\mathcal{F}_2}$ at time t_2. Both poses have the same reference frame $\mathcal{F}_0$. We can split up the poses in their orientations $\boldsymbol{R}_1, \boldsymbol{R}_2$ and positions $\boldsymbol{t}_1, \boldsymbol{t}_2$, respectively.

First, we discuss interpolation of position. In the simplest case, besides the positions $\boldsymbol{t}_1, \boldsymbol{t}_2$ at times t_1, t_2, no additional information is given. Therefore, we can only guess how the object moves between the positions $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$. We assume a constant velocity vector as an intuitive motion model. This results in linear interpolation for the position $\boldsymbol{t}_t$ at time $t \in [t_1, t_2]$:

$$\boldsymbol{t}_t = \boldsymbol{t}_1 + \frac{t - t_1}{t_2 - t_1}(\boldsymbol{t}_2 - \boldsymbol{t}_1) \quad . \tag{3.66}$$

Linear interpolation of positions is a common method because the assumption of a constant velocity vector is often reasonable and it is efficient to calculate. The downside of linear interpolation is that when dealing with more than just two positions, interpolation leads to a jerky trajectory. Often, we assume that objects move smoothly. Splines (see subsection 3.4.2) can be used to represent a smooth movement in 3D space. To interpolate between two positions based on a spline, the spline has to be fit to all positions in the relevant timespan.

This can be done with least-squares optimization (see subsection 3.5.2). Then, the control parameter of the time for interpolation has to be determined and the spline has to be evaluated at this value. We can also represent linear interpolation by a simple spline constructed by B-splines of degree one, knot vector $[0, 0, 1, 1]$ and control points $\boldsymbol{t}_1, \boldsymbol{t}_2$.

Interpolation of orientation is not as straightforward as for position. A simple attempt would be to linearly interpolate rotation matrices. This does not work because the resulting matrix is in many cases not even in $\boldsymbol{SO}(3)$ anymore, and therefore, does not represent an orientation. Another attempt would be to work with Euler angles. Linear interpolation of the three angles leads to unintuitive interpolation with changing rotation rate [DKL98].
Interpolation of orientations is mostly performed in the quaternion space. To understand why, we need to discuss quaternions once more:
Unit quaternions can be represented as 4D vectors with norm one. So, we can think of a unit quaternion as a point on the surface of a 4D unit sphere. Every point on this unit sphere corresponds to an orientation. One can show that moving a distance x on the surface of the unit sphere corresponds to a rotation of $2x$, e.g. an arc of $90\,^\circ$ on the sphere corresponds to a rotation of $180\,^\circ$. An intuitive interpolation between two orientations is to take the shortest path possible without any detour. This corresponds to moving on a great circle. A great circle is the intersection of the sphere with a hyperplane that intersects the center of the sphere. Additionally, we want to have a constant rate of rotation which corresponds to a constant velocity on the sphere. So, to summarize, intuitive interpolation for orientations corresponds to moving with constant velocity on a great circle from one 4D unit quaternion vector to the other. This kind of interpolation is called *spherical linear interpolation* and is often abbreviated by *Slerp*. Slerp is not specific to interpolating rotations in quaternion space. It is a general concept which can be applied to unit vectors in any dimension. The algorithm for Slerp is independent of the dimension. In the following, we derive Slerp in 2D:
Let's assume we want to interpolate between two unit vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. The angle θ_0 between the vectors can be calculated by $\theta_0 = \arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)$. The angle between the interpolated vector $\boldsymbol{v}_t$ and $\boldsymbol{v}_1$ is $\theta = t\theta_0$, with the interpolation parameter $t \in [0, 1]$. Figure 3.10 a) visualizes the vectors on the unit circle. The goal is to find an expression of $\boldsymbol{v}_t$ in terms of $t, \boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Let's

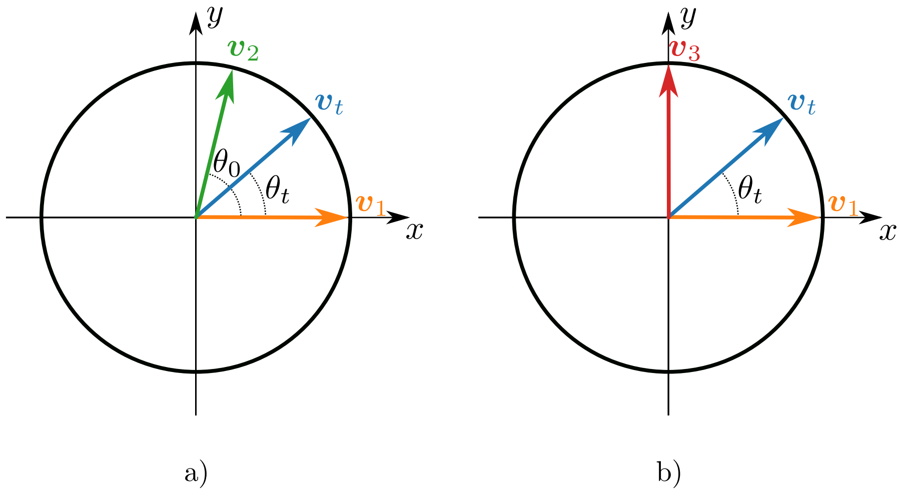


Figure 3.10: Sketch to derive Slerp in 2D. The unit sphere in 2D is the unit circle. a) Unit circle with interpolated vector $\boldsymbol{v}_t$ between vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. b) Same as a) but $\boldsymbol{v}_2$ is replaced by $\boldsymbol{v}_3$ that is orthogonal on $\boldsymbol{v}_1$.

first consider the special case that $\boldsymbol{v}_2$ is replaced by $\boldsymbol{v}_3$ which is orthogonal to $\boldsymbol{v}_1$ (see Figure 3.10 b). For this special case, simple trigonometry leads to

$$\boldsymbol{v}_t = \cos(\theta_t)\boldsymbol{v}_1 + \sin(\theta_t)\boldsymbol{v}_3 \tag{3.67}$$

$$= \cos(t\theta_0)\boldsymbol{v}_1 + \sin(t\theta_0)\boldsymbol{v}_3 \quad . \tag{3.68}$$

If we assume that $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are not parallel, we can generate a vector $\hat{\boldsymbol{v}}_3$ which is orthogonal on $\boldsymbol{v}_1$ and the corresponding normed vector $\boldsymbol{v}_3$:

$$\hat{\boldsymbol{v}}_3 = \boldsymbol{v}_2 - (\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)\boldsymbol{v}_1 \tag{3.69}$$

$$\boldsymbol{v}_3 = \frac{\hat{\boldsymbol{v}}_3}{\|\hat{\boldsymbol{v}}_3\|} \quad . \tag{3.70}$$

We can combine these results in a simple algorithm which calculates Slerp for any dimension (see Algorithm 1). This algorithm can be compressed in one formula [Bus03]:

$$\mathrm{Slerp}(\boldsymbol{v}_1, \boldsymbol{v}_2, t) = \frac{\sin[(1-t)\arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)]}{\sin[\arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)]}\boldsymbol{v}_1 + \frac{\sin[t\arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)]}{\sin[\arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)]}\boldsymbol{v}_2 \tag{3.71}$$

$$= \frac{\sin[(1-t)\theta_0]}{\sin[\theta_0]}\boldsymbol{v}_1 + \frac{\sin[t\theta_0]}{\sin[\theta_0]}\boldsymbol{v}_2 \quad . \tag{3.72}$$

Algorithm 1 : Spherical linear interpolation (Slerp)

```
1 Slerp ($\boldsymbol{v}_1, \boldsymbol{v}_2, t$);
  Input  : Start and end vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ and interpolation
           parameter $t$.
  Output : Interpolated vector $\boldsymbol{v}_t$.
2 $\theta_0 = \arccos(\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)$;
3 $\theta_t = t\theta_0$;
4 $\boldsymbol{v}_3 = \boldsymbol{v}_2 - (\boldsymbol{v}_1 \cdot \boldsymbol{v}_2)\boldsymbol{v}_1$;
5 $\boldsymbol{v}_3 = \mathrm{normalize}(\boldsymbol{v}_3)$;
6 $\boldsymbol{v}_t = \cos(\theta_t)\boldsymbol{v}_1 + \sin(\theta_t)\boldsymbol{v}_3$;
7 return $\boldsymbol{v}_t$
```

For interpolating orientations based on quaternions, we have to perform an additional check before using Slerp. Because a quaternion $\boldsymbol{q}$ represents the same orientation as $-\boldsymbol{q}$ we have to check if the signs in the interpolation pair $\boldsymbol{q}_1, \boldsymbol{q}_2$ are correct. If $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ have a distance of more than $180\,^\circ$ on the unit 3-sphere (remember that this would be an interpolation path of more than $360\,^\circ$), we need to switch the sign of one of the quaternions.

Slerp is by far the most commonly used method to interpolate orientations. For small rotations between orientations, a very fast approximation of Slerp is the so-called *normalized linear interpolation* (*NLerp*). It is standard linear interpolation of two quaternion vectors but with following normalization to end up with a unit quaternion again. NLerp can be derived from Algorithmus 1 or Equation 3.72 with small angle approximations.

3.5 Model Fitting

Model fitting is the task of finding a model which fits given data. Having a model which describes the data well is a powerful tool that enables efficient data processing and often leads to a better understanding of the data. That makes model fitting an essential task in a great variety of fields e.g. image processing, finance and marketing. In the following, we present two common concepts which can be used to fit a model to data.

3.5.1 Random Sample Consensus

Random Sample Consensus (RANSAC) is a sampling-based method to find a model that fits given data. RANSAC is best known for its capability to fit models to data with a high number of outliers. The following details on RANSAC are based on the original publication [FB81]:

Let P be the set of m given data points for which a model M should be fitted. To instantiate the model M, a minimum of n data points is needed. We assume that $m \geq n$.
RANSAC iteratively samples a subset S_i of P with n points. Based on this subset S_i, a model M_i is instantiated. A subset S_i^* is determined by isolating points in P which support the model M_i within some tolerances. We call S_i^* the consensus set of S_i. A preset threshold τ is used to reject or accept the model M_i as a valid solution. If the number of points in the consensus set S_i^* is equal to or higher than τ, the model is accepted and the algorithm is terminated. If the number of points in S_i^* is lower than τ, the model is dismissed and a new subset S_{i+1} is randomly sampled. The process continues till a valid model is found or the maximum number of allowed iterations is reached.

Several parameters have to be set to use the RANSAC algorithm:
To determine a consensus set S_i^*, we need to set the error tolerance which defines whether a point is an inlier or an outlier with respect to the model M_i. Usually, this parameter has to represent the requirements of the application, and therefore, no fixed rule to set this parameter can be given. Also τ depends on the application and is often determined experimentally. As a rule of thumb, a good threshold to start with is $\tau = \omega m$, where ω is the inlier ratio of P.

Usually, ω is not known precisely but can be roughly estimated.
The last parameter to set is the number of iterations k. Let p be the desired minimum probability of finding a valid model. A valid model is picked if all n samples to instantiate the model are inliers. Let's assume that all n samples are selected independently from P (valid assumption for a large dataset P). Then, ω^n is the probability that all n points are inliers. The probability that at least one of the n points is an outlier can be calculated by $1 - \omega^n$. So, in a single try, an invalid model is selected with probability $1 - \omega^n$. Consequently, no valid model is found in k trials with probability $(1 - \omega^n)^k$. This probability is limited by the desired maximum probability of not finding a valid model which is $1 - p$:

$$(1 - \omega^n)^k \leq 1 - p \quad . \tag{3.73}$$

Based on this relation, we deduce the minimum number of trials:

$$k = \frac{\log(1 - p)}{\log(1 - \omega^n)} \quad . \tag{3.74}$$

The RANSAC algorithm determines a model M_i only based on n data points. The rest of the data is only used to validate the model. Therefore, the authors of [FB81] suggest to perform additional optimization of the model based on all inlier points (see subsection 3.5.2).
MSAC (M-Estimator Sample Consensus) [TZ00], which is a generalization of RANSAC, introduces a more sophisticated way to determine how well a model fits the data. Instead of only using an error threshold which is used to classify a point into inlier or outlier, MSAC uses an error function which allows to encode more accurately how well the model fits the data. Usually, this results in finding a model which fits the data better than with the original RANSAC algorithm.

3.5.2 Least-Squares Optimization

Least-squares optimization is a very common method for model fitting. It is popular because of its ability to conveniently solve a wide range of practical problems in an efficient way.

Let's assume we have a model parameterized by the vector $\boldsymbol{x} = [x_1, ..., x_n]^T$. We define some smooth functions $r_1(\boldsymbol{x}), ..., r_m(\boldsymbol{x})$ that measure how well the model fits some data. We call these functions residuals and combine all residuals in a residual vector $\boldsymbol{r}(\boldsymbol{x}) = [r_1(\boldsymbol{x}), ..., r_m(\boldsymbol{x})]^T$. The main idea of least-squares optimization is to find the parameter vector $\boldsymbol{x}^*$ that minimizes the sum of squared residuals

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \frac{1}{2} \sum_{i=1}^{m} r_i^2(\boldsymbol{x}) \tag{3.75}$$

$$= \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{r}(\boldsymbol{x})\|_2^2 \ . \tag{3.76}$$

In the following, we assume that $m \geq n$.

There are multiple reasons for minimizing the squared Euclidean norm and not e.g. the absolute value of the residuals. When using the squared Euclidean norm of the residuals, the structure of the minimization problem allows for useful approximations which increase efficiency significantly. Another reason is that there are many instances in which minimizing the squared Euclidean norm makes good statistical sense. One important example is the case when the residuals are assumed to be independent from each other and distributed with the identical normal distribution. Then the least-squares solution is a maximum likelihood estimate.

Linear Least-Squares

Let $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ be a matrix and $\boldsymbol{y} \in \mathbb{R}^m$ a vector. A least-squares problem with a residual vector $\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{J}\boldsymbol{x} - \boldsymbol{y}$ is called *linear least-squares problem*:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{J}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \ . \tag{3.77}$$

One can show that Equation 3.77 is a convex optimization problem, and therefore, any point $\boldsymbol{x}^*$ with

$$\nabla \left(\frac{1}{2} \|\boldsymbol{J}\boldsymbol{x}^* - \boldsymbol{y}\|_2^2 \right) = \boldsymbol{0} \tag{3.78}$$

is a solution of Equation 3.77 ($\boldsymbol{x}^*$ is a global minimizer) [NW06]. Simple algebra leads to

$$\boldsymbol{J}^T \boldsymbol{J}\boldsymbol{x}^* = \boldsymbol{J}^T \boldsymbol{y} \ \ , \tag{3.79}$$

which is a system of linear equations. These equations are called *normal equations* for the problem in Equation 3.77 because $\boldsymbol{J}\boldsymbol{x}^* - \boldsymbol{y}$ is normal to the range of $\boldsymbol{J}$ (more obvious in the form $\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{x}^* - \boldsymbol{y}) = \boldsymbol{0}$).
The normal equations are usually solved by Cholesky factorization, QR factorization or singular value decomposition (SVD). Each of the methods has its advantages and disadvantages [NW06]: Cholesky factorization is useful if $m \gg n$ and $\boldsymbol{J}$ is sparse but cannot be used straightforward if $\boldsymbol{J}$ is rank-deficient or ill conditioned. QR factorization is more robust numerically. SVD is the most robust and reliable approach but also the most computationally expensive. For very large problems, it might be more efficient to use an iterative method.

Nonlinear Least-Squares

Nonlinear least-squares problems are nonlinear in the residual vector $\boldsymbol{r}(\boldsymbol{x})$. To solve nonlinear least-squares problems, iterative solving methods are used.

One of them is the *Gauss-Newton method*: The main idea of this method is to approximate the residual vector $\boldsymbol{r}(\boldsymbol{x})$ linearly at the current solution $\boldsymbol{x}_k$ which is

$$\boldsymbol{r}(\boldsymbol{x}) \approx \boldsymbol{r}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) \ \ , \tag{3.80}$$

with the Jacobian matrix

$$\boldsymbol{J}(\boldsymbol{x}) = \left[\frac{\partial r_i}{\partial x_j}\right]_{\substack{i=1,\ldots,m\\ j=1,\ldots,n}} = \begin{pmatrix} \nabla r_1(\boldsymbol{x})^T \\ \nabla r_2(\boldsymbol{x})^T \\ \vdots \\ \nabla r_m(\boldsymbol{x})^T \end{pmatrix} \quad . \tag{3.81}$$

So, at each iteration, we solve a linear least-squares problem for the correction

$$\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} \frac{1}{2}\|\boldsymbol{r}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{p}\|_2^2 \tag{3.82}$$

with the methods explained before. We update our solution by $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k$. The Gauss-Newton method converges very quickly for mildly nonlinear residuals (one step for the linear case). The downside of this method is that if we start far from the solution or the residual is highly nonlinear then the method might not even converge locally [Bjö96].

The *damped Gauss-Newton method* alleviates the problems of the Gauss-Newton method. The update step is modified by introducing a step length α_k:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \quad . \tag{3.83}$$

The search direction $\boldsymbol{p}_k$ is still calculated based on Equation 3.82. The damped Gauss-Newton method is said to be a *line search method*. At each interation, the objective function is reduced along a line with direction $\boldsymbol{p}_k$. There are multiple ways to find an appropriate step size α_k. A simple but common way is to start at $\alpha_k = 1$ and halving as long as

$$\|\boldsymbol{r}(\boldsymbol{x}_k)\|_2^2 - \|\boldsymbol{r}(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)\|_2^2 \geq \frac{1}{2}\alpha_k \|\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{p}_k\|_2^2 \tag{3.84}$$

does not hold anymore [Bjö96]. This inequality is an intuitive measure of how well the linearization approximates the non-linear residual vector for the step size of α_k along the step direction $\boldsymbol{p}_k$. Another common method to chose the step size α_k is to solve the one dimensional optimization problem

$$\alpha_k = \arg\min_{\alpha} \frac{1}{2} \|\boldsymbol{r}(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)\|_2^2 \ . \tag{3.85}$$

The standard and the damped version of the Gauss-Newton method can encounter problems when the Jacobian matrix at one iteration does not have full column rank, or nearly so [EG04, NW06]. The *Levenberg-Marquardt method* solves this problem and is therefore more stable than Gauss-Newton methods. It can be iteratively formulated as the solution of the constraint problem

$$\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} \frac{1}{2} \|\boldsymbol{r}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{p}\|_2^2 , \quad \text{subject to } \|\boldsymbol{p}\| \leq \Delta_k \ , \tag{3.86}$$

where $\Delta_k > 0$. The Levenberg-Marquardt method is called a *trust region method* and Δ_k is the *trust region radius*. The interpretation is that the linearization $\boldsymbol{r}(x) \approx \boldsymbol{r}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$ is trusted only for $\|\boldsymbol{x} - \boldsymbol{x}_k\| \leq \Delta_k$. The constrained problem of Equation 3.86 can be reformulated as the regularized problem

$$\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} \frac{1}{2} \|\boldsymbol{r}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{p}\|_2^2 + \mu_k \|\boldsymbol{p}\|_2^2 \ , \tag{3.87}$$

with $\mu_k \geq 0$. We can write this as a standard linear least-squares problem

$$\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} \frac{1}{2} \left\| \begin{pmatrix} \boldsymbol{J}(\boldsymbol{x}_k) \\ \sqrt{\mu_k}\boldsymbol{I} \end{pmatrix} \boldsymbol{p} + \begin{pmatrix} \boldsymbol{r}(\boldsymbol{x}_k) \\ \boldsymbol{0} \end{pmatrix} \right\|_2^2 \tag{3.88}$$

that is easy to solve with the methods mentioned before.
For practical cases, the spherical trust region in Equation 3.86 is modified to an ellipsoidal trust region to account for the characteristics (e.g. curvature and range) for each dimension in $\boldsymbol{x}$. Different scales can lead to numerical problems and slow convergence with a spherical trust region [NW06]. Mathematically, the subproblem at each iteration can be formulated as

$$\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} \frac{1}{2} \|\boldsymbol{r}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{p}\|_2^2 , \quad \text{subject to } \|\boldsymbol{D}_k \boldsymbol{p}\| \leq \Delta_k \ , \tag{3.89}$$

where the diagonal matrix $\boldsymbol{D}_k$ has positive diagonal entries. This can be solved as a linear least-squares problem

$$\boldsymbol{p}_k = \arg\min_{\boldsymbol{p}} \frac{1}{2}\left\| \begin{pmatrix} \boldsymbol{J}(\boldsymbol{x}_k) \\ \sqrt{\mu_k}\boldsymbol{D}_k \end{pmatrix} \boldsymbol{p} + \begin{pmatrix} \boldsymbol{r}(\boldsymbol{x}_k) \\ \boldsymbol{0} \end{pmatrix} \right\|_2^2 . \tag{3.90}$$

One question is still remaining: How do we set $\boldsymbol{D}_k$ and Δ_k?
The diagonal matrix $\boldsymbol{D}_k$ can be set to incorporate the information about the different gradients for the dimensions of $\boldsymbol{p}$. If the gradient in one dimension is small, we want to allow for larger steps in this dimension to prevent slow convergence. This is accomplished by a low weighting for this dimension through the according diagonal element in $\boldsymbol{D}_k$. A common way is to set the diagonal elements of $\boldsymbol{D}^2$ to the same as the ones in $\boldsymbol{J}_k^T \boldsymbol{J}_k$.
The trust region radius is iteratively corrected. The initial radius Δ_0 is set high. The radius is reduced if the discrepancy between the linearization and the nonlinear function is too high. The measure

$$\rho_k = \frac{\|\boldsymbol{r}(\boldsymbol{x}_k)\|_2^2 - \|\boldsymbol{r}(\boldsymbol{x}_k + \boldsymbol{p}_k)\|_2^2}{\|\boldsymbol{r}(\boldsymbol{x}_k)\|_2^2 - \|\boldsymbol{r}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{p}_k\|_2^2} \tag{3.91}$$

is used to assess the calculated update proposal $\boldsymbol{p}_k$. An iteration is said to be successful if $\rho_k > \beta$ with a fixed $\beta \in (0, 1)$ [Bjö96]. Then, we update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k$. The trust region is increased for a successful iteration e.g. by two. If the iteration is not successful, we set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and reduce the trust region e.g. by half.

Robustification

In many real scenarios, the data at hand is not free of outliers. Least-squares fits are sensitive to outliers. The severity of the outliers and the outliers-to-inliers ratio decide how much the estimation quality suffers. There are different approaches to detect and handle outliers:

For data with a high ratio of outliers to inliers, sampling based methods are suitable. RANSAC (see subsection 3.5.1) can be used to isolate inliers, and then, least-squares optimization can be used on the inlier subset.

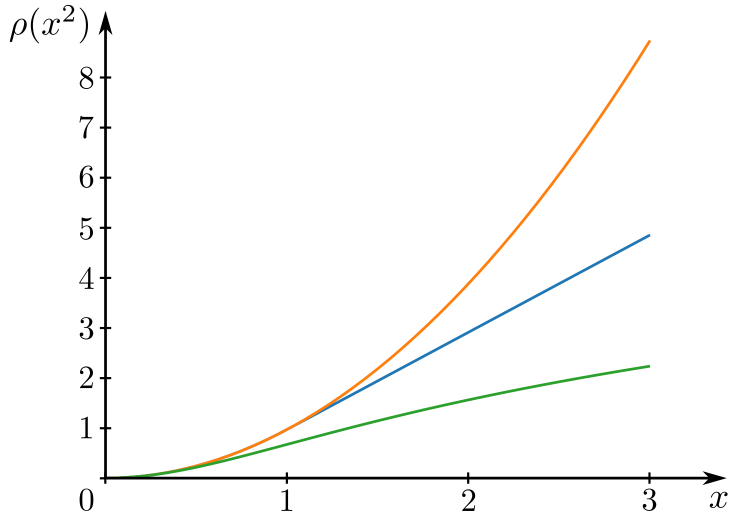


Figure 3.11: Comparison of trivial loss (orange), Huber loss (blue) and Cauchy loss (green).

Least-squares problems are very sensitive to outliers because squaring high residuals results in very high costs. As a consequence, a few outliers can dominate the problem and lead to bad results. If there are only few outliers in the data, a simple but effective method for robustification is to weight the squared residuals by so-called *loss functions* ρ. Loss functions are scalar functions $\rho \colon \mathbb{R} \to \mathbb{R}$ that downweight high residuals to reduce their influence on the solution. Especially after some solving iterations, these high residuals usually correspond to outliers in the data. The modified least-squares problem can be formulated as

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \frac{1}{2} \sum_{i=1}^{m} \rho_i\left(r_i^2(\boldsymbol{x})\right) \quad . \tag{3.92}$$

For the special case $\rho_{\text{trivial}}(s) = s$ (trivial loss), this is a standard least-squares problem. Multiple types of loss functions are commonly used. In Figure 3.11, the Huber loss and the Cauchy loss are compared to the trivial loss. Mathematically, these loss functions are defined as

$$\rho_{\text{Huber}}(s) = \begin{cases} s & \text{if } s \leq 1 \\ 2\sqrt{s} - 1 & \text{else} \end{cases} \tag{3.93}$$

$$\rho_{\text{Cauchy}}(s) = \log(1 + s) \quad . \tag{3.94}$$

They are designed to resemble the trivial loss for $s < 1$ and to be well below the trivial loss for $s \gg 1$. Often, residuals are differently scaled, e.g. a residual of 0.1 might be considered high or a residual of 10 might be considered low. Therefore, normalization of the residuals is recommended. In this way, high residuals ($s \gg 1$) are downweighted while low residuals ($s < 1$) are not.

When using a loss function, the optimization is not a least-squares problem anymore. One popular method to solve the problem is *iteratively reweighted least-squares* (IRLS). The problem is reformulated as a least-squares problem with weights for the original residuals. The weights are recalculated for each iteration.

4 Calibration Target

For all our calibration methods presented in this work, we make use of a calibration target. As discussed in chapter 2, calibration methods that use a known calibration target have several advantages over targetless methods:
A target provides invariance to the scene such that for all environments similar calibration results can be expected. Because the target is well known, outlier rejection is simple and reliable which is essential for high quality results. Most importantly, an ideal target is designed to generate very accurate correspondences between the sensors.
Targetless methods are highly dependent on the scene, and therefore, are not as reliable as calibrating with a known target. Further, usually strong initialization is needed and often only a subset of all setup parameters can be estimated.

In chapter 2, we introduced and discussed many calibration targets that were proposed in the last decades. Based on their advantages and disadvantages, we define criteria that our calibration target should fulfill:
First, we aim for fully automatic calibration. Many methods need user input for segmenting the target from the scene, marking features or solving an association problem. Involving the user makes the method inconvenient and the results are not independent of the user anymore. So, no user input should be required. Therefore, the calibration target should be unique in the environment so that it can be easily segmented. Especially planes are omnipresent so that it is not trivial to automatically segment a planar calibration target in 3D point cloud data. Known dimensions of the board ease the problem but still consistency checks must be applied since a single undetected outlier can significantly reduce the quality of the calibration. In camera data, uniqueness can be created by using patterns or tags (see subsection 3.2.3). For radars, a target can be designed to have a very high RCS that usually does not occur in a scene, and therefore, makes the target unique.
After the target is segmented, features that can be accurately located in data of all sensors have to be detected. For 3D point cloud data, the measurements

should sample the features accurately without limitation of the sensor's resolution. E.g. measurements that fall on a planar board sample a plane or measurements that fall on a spherical target sample a sphere. In both cases, if the measurements are not noisy, the plane or sphere can be perfectly estimated, no matter where the measurements hit. In contrast to these examples, 3D corners or edge features that are estimated based on range discontinuities cannot be estimated perfectly because of the limited resolution of the sensor. Many approaches presented in literature use corners and edges for 3D point cloud data (see chapter 2), and therefore, systematically limit their feature detection accuracy which is essential for high quality calibration results. So, for 3D point cloud data, surface features that can, theoretically, be estimated perfectly should be used.

For cameras, high contrast edges, or even better, corners should be used. Prints with patterns are very convenient to create, provide high accuracy and guarantee high contrast (if lighting is reasonable). If the edges or corners result from an object occluding another, the foreground and background should be controlled to ensure that the contrast is high. E.g. a white board in front of a white wall will not work, whereas a black board in front of a white wall provides high contrast and can therefore be detected with high accuracy.

Printing patterns on a board is the most convenient approach to create features for cameras on the target but different colors have different reflectivity which causes range errors in LiDAR data [PYW$^+$14]. The severity of this problem depends on the LiDAR model but only very few do not suffer from it. So, features for LiDAR data are ideally derived from surfaces with homogeneous reflectivity.

In general, it is beneficial if many features are connected. E.g. the pose of a checkerboard can be more accurately determined in camera if it has many corners on it or the normal of a board can be more accurately estimated in 3D point cloud data if more measurements fall on the board. Therefore, we usually want the target to be large. In contrast to this, we want the target to be portable and lightweight. Often, it is inconvenient or even impossible to move the sensor setup, so that the target has to be moved around the sensors and not the other way round. So, a compromise has to be made between a large target, that is observed by many measurements but is heavy and not very portable, and a small target, that is observed by less measurements but is lightweight and can be moved around easily.

After segmenting the target and detecting features on it, many methods have to solve an association problem because the target has multiple identical features on it. Again, we do not want to resolve the association problem by the user. A simple way to prevent this problem is to only use a single feature. Another approach is to use additional information (e.g. additional markers or shapes) or to make the features unique.

We propose to use a spherical calibration target. In its simplest version, it is a white Styrofoam sphere with an attached stick to hold it. The sphere is an off-the-shelf product which can be purchased in many hardware stores. It has a diameter of 50 cm with a roundness that deviates less than a millimeter. Because it is hollow, the sphere is lightweight with less than 300 g which makes it convenient to move around. A sphere shows very handy mathematical properties. It is modeled by only its center point and its radius. No orientation has to be estimated. We will later see that 3D residuals can be defined easily for all sensors and can be efficiently calculated.
A sphere of this size is usually unique in a scene. It is easy to segment in 3D point cloud data. With a diameter of 50 cm, it is large enough to be detectable in decent ranges even with low resolution LiDARs. The white sphere has a homogeneous surface so that no problems due to varying reflectivity occur. High detection accuracies can be reached in 3D point cloud data because the estimation accuracy is not limited by the resolution of the sensor.
In image data, the sphere can be localized based on the edge contour between sphere and background. So, the accuracy of the edge detection is critical for the calibration quality. For high accuracy edge detections, the contrast between sphere and background must be high. Therefore, the simples version of our sphere target can only be used for camera calibration if the background is significantly darker than the sphere. For best results, we use an additional background which is attached to the target. The background is created from a special light absorbing fabric. The downside of using a background is that it adds weight to the target and the target cannot be detected from any angle anymore. In case of bad lighting, we can increase contrast by turning on a light that is mounted inside of the sphere. The Styrofoam with a wall thickness of 2 cm lets light pass through. To make the sphere segmentation faster and more robust in camera images, ArUco markers and lines are attached to the background (see Figure 4.1).

A Styrofoam sphere cannot be detected by automotive radars. Therefore, we make use of a corner reflector which is placed in the inside of the sphere (see Figure 4.2). The corner reflector is build from metal sheets that reflect the

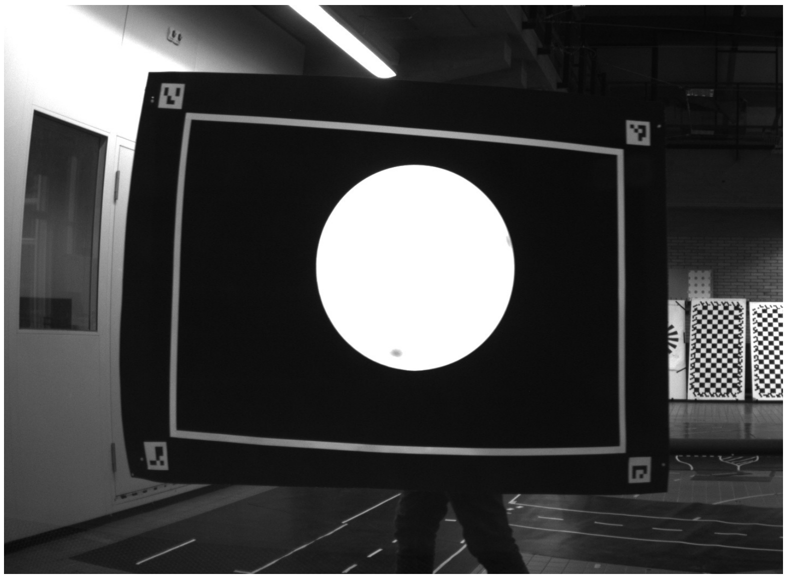


Figure 4.1: Calibration target with background. A light absorbing fabric in combination with LED lighting in the inside of the sphere provides high contrast between the background and the sphere.

radar signal. The metal sheets are attached orthogonally to each other. Due to this geometry, the signals are reflected back in the direction from which they come.
The RCS of a corner reflector is very large for its size. Its maximum can be calculated by

$$\sigma = \frac{4\pi a^4}{3\lambda^2} \ , \tag{4.1}$$

where a is the length of the side edges of the three isosceles triangles (see Figure 4.3) and λ is the wavelength of the signal. This formula is valid for signals that hit the three metal sheets with equal incident angles. The RCS decreases the more it differs from this angle because an increasing part of the signal is not reflected back to the source (see Figure 4.4). To segment the corner reflector from all other reflecting objects in the scene, we want its RCS to be higher than the RCS of any other object. From Equation 4.1, we see that the RCS is proportional to the fourth power of the edge length of the triangular sheets. Hence, we want to maximize the size of the triangles.
Ideally, the corner reflector is detected as an effective point at the center of the sphere. In the following, we answer the question where the effective detection

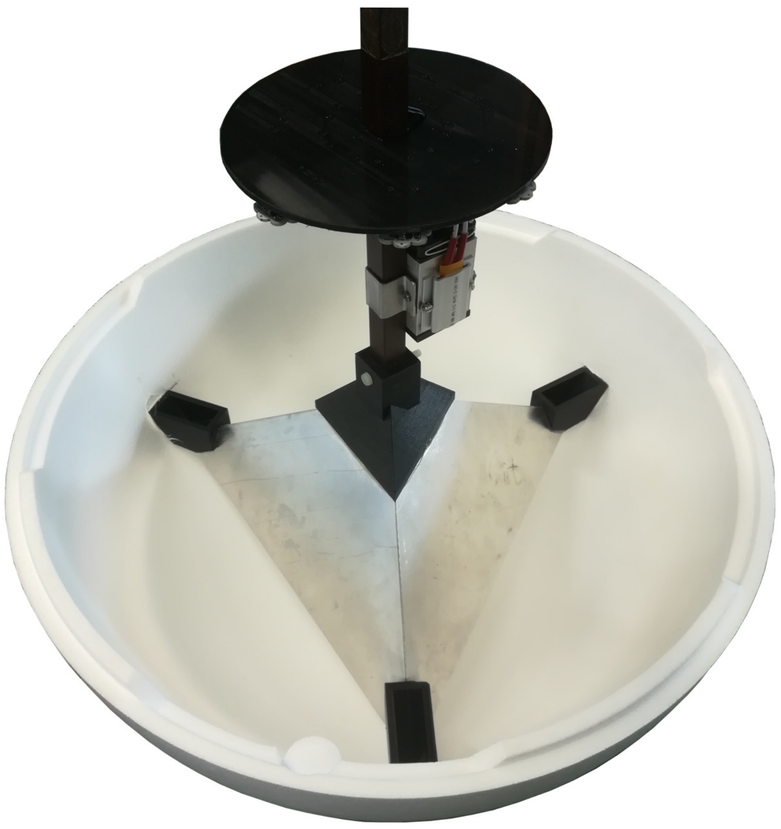


Figure 4.2: One half of the Styrofoam sphere with the metal corner reflector attached to it. Additionally, a heater and LEDs are placed inside the sphere.

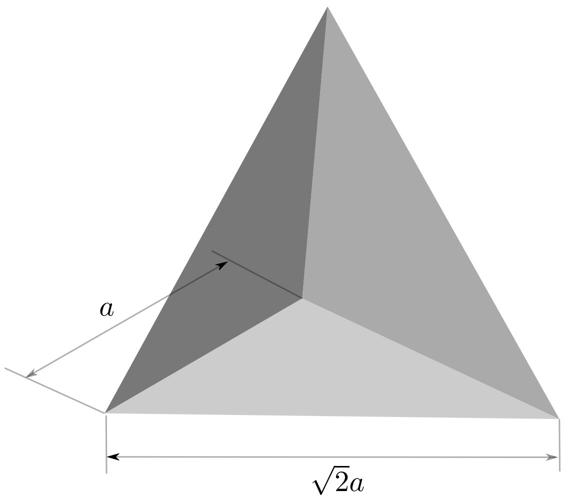


Figure 4.3: Sketch of a 3D corner reflector.

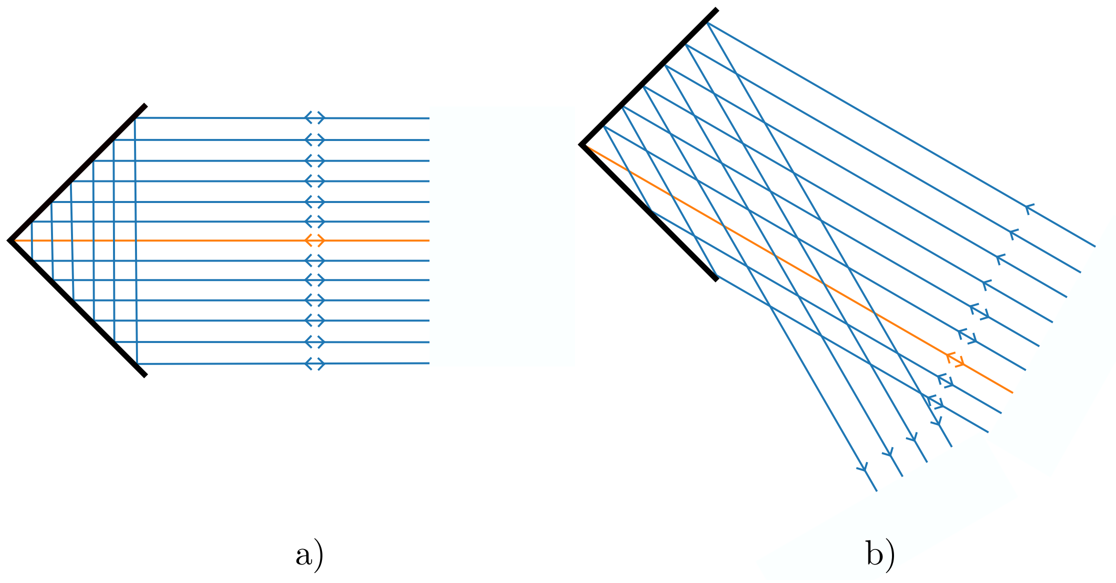


Figure 4.4: Reflection patterns of a 2D corner reflector for two different incident angles. In a), the incident angle is 45 ° at both sides. All the incoming rays are reflected at both sides and return in the direction they come from. In b), the incident angle for each side is different. Not all rays are reflected at both sides, and therefore, not all rays return to the source.

point of a corner reflector is and if it depends on the incident angle of the signal. As a simplification, we derive the effective detection point for the 2D case since it is better for visualization. The derivative can directly be transfered to the 3D case. Further, we use the concept of ray geometry from optics.
Let's assume a ray hits the corner reflector with an angle of α (see Figure 4.5). The ray is reflected on both sides of the corner reflector and returns parallel to the initial ray. In reality, a radar signal is not as focused as a single ray but hits the corner reflector over its full width. On average, the radar detects the corner reflector in the direction of the ray that passes through the corner of the reflector (see the orange ray in Figure 4.5). Based on the time of flight, a radar measures the distance to a surface. The remaining question is: at which effective range is the corner reflector detected?
The range measurement can be calculated by the length of the path which the ray travels. We show that the range measurement of the blue ray in Figure 4.5 is the same as for the ray that hits the reflector in the corner. Up to the dashed line, the traveled distance is the same. The distances for the rest of the rays can

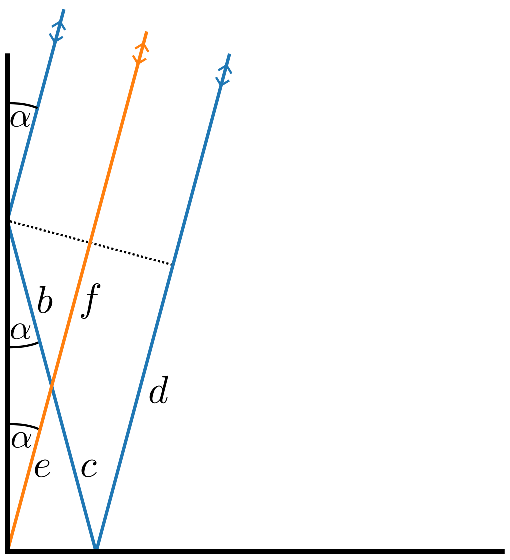


Figure 4.5: 2D corner reflector and different rays. The orange ray hits exactly the corner of the reflector.

be calculated as $b + c + d$ for the blue ray and $2(e + f)$ for the orange ray. The difference between these two distances is zero:

$$\Delta = [b + c + d] - [2(e + f)] \tag{4.2}$$
$$= [e + e + d] - [2(e + f)] \tag{4.3}$$
$$= d - 2f \tag{4.4}$$
$$= 0 \tag{4.5}$$

For Equation 4.3, we use that $b = e$ and $c = e$ because b and c are in isosceles triangles with e. Finally, we use the rules for similar triangles to find $d = 2f$. The interpretation of this result is that the corner reflector is detected effectively at the range of the corner point. So, the effective point measurement of the corner reflector is its corner point. Therefore, we place the corner of the reflector at the center of the sphere.

The maximal size of the corner reflector can be easily determined by noticing that the short edges of the triangles must be the inner radius R_{inner} of the sphere. In our case, that is $a = R_{\text{inner}} = 23$ cm. The resulting RCS is about 13 times the RCS of an average human. Usually, only few static object have an RCS equal or higher than this corner reflector. These static objects can be automatically filtered out (see section 5.3)

Finally, we also mount a heater in the inside of the sphere to control its temperature. This shall be used for calibration of a thermal camera e.g. for applications that fuse thermal and texture information, such as in [3]. Because the sphere is not connected to the background, the thermal difference between background and sphere is large. The sphere diameter expands by 0.6 mm for a temperature increase of 15 °C, which should be taken into account.

5 Target Detectors

In this chapter, we introduce methods for detecting the calibration target in camera, LiDAR and radar data.

At this point, we want to clarify the use of the terms *measurement*, *observation* and *detection*. The terms observation and detection both address the observed/detected center position of the spherical calibration target. The term measurement is used in the sense of a single data point from a sensor. For LiDAR, measurements are 3D points. For camera, measurements are 2D points in image coordinates. For radar, measurements are 2D points in the radar scan plane. Observations can be calculated from measurements. E.g. multiple 3D points (measurements) from a LiDAR that sample the sphere can be used to derive the center point of the sphere (observation).

5.1 Camera Sphere Detector

For detecting a projected sphere in an image, we differentiate between multiple cases:
In the most general case, the detector is applied to raw images which are not corrected for distortions. Theoretically, the sphere can have any shape in the image. In a simpler case, the distortions are corrected and a pinhole model is used for the camera. Then, the projection of a sphere, essentially, is the intersection between the image plane and the viewing cone from the optical center to the sphere (see Figure 5.1 a). It can be shown that the intersection is an ellipse. In the simplest case, the distortions are corrected and a spherical camera model is used. The intersection between the spherical projection screen and the cone to the spherical target is a circle (see Figure 5.1 b). As explained in subsection 3.2.2, a circle can be described by less parameters than an ellipse and is therefore easier to detect in an image.

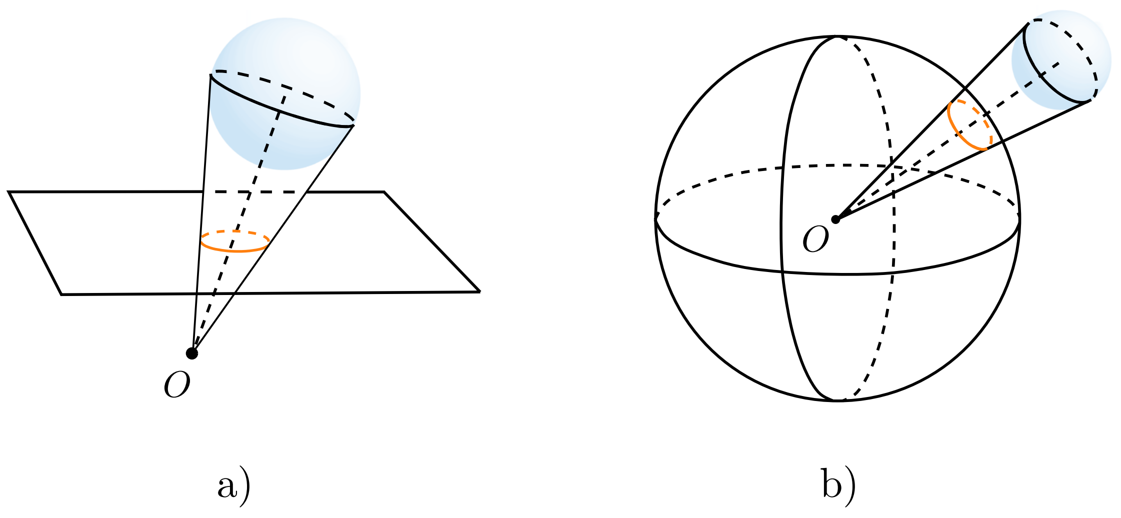


Figure 5.1: Viewing cones from optical center O to the spherical calibration target (blue). In a), the sphere is projected on a plane. In b), the sphere is projected on a sphere. The projections are drawn in orange.

Spherical Camera Model

We first discuss the implemented sphere detector for the case that a spherical camera model is used, hence, the projection is a circle:
In most calibration frameworks, a dataset is recorded first and then the detection algorithm is run on this dataset. This approach suffers from two drawbacks. First, a full assessment of the lighting and camera settings is only possible when the detection algorithm is run. When detecting the target after recording the dataset, a correction of the settings also means that the recording has to be repeated. Second, at recording time, we do not know the number and distribution of detections. So, we do not know if enough data is recorded. The solution is to have a detector that can be run in real time while recording. This allows to check if the calibration target can be detected with the current settings and enables monitoring the distribution and number of detections. To detect the target in real time, the detection algorithm must be very efficient. To reach high efficiency, we use the assumption that the camera is static and the sphere is moving. In the beginning of the calibration process, we detect edges in the static scene over multiple images (e.g. 10 frames). Edges that consistently occur are stored. We refer to them as static edges.
For each new image, we run our edge detector and remove the static edges. The resulting edges are only on moving objects. If only the sphere and the

person who carries the sphere are moving then the amount of edges is very small. Therefore, subtracting the static edges is an early filtering stage that is essential for high efficiency and reliability.
The few remaining edges are passed to circle Hough Transform (see subsection 3.2.2). The resulting circles are initial candidates and must be filtered and refined in the following process. Two strong filter criteria are used to reduce the candidate count. The first criterion assesses completeness of the circle support. If edge detections are all around the circle, we say that the circle has complete support. A strong candidate must have edge detections at more than 50 % of the expected circle pixels. The second filtering criterion is used to prevent that a circle is fitted on a random accumulation of edge pixels. The edge detections in the neighborhood around the circle are counted and compared to the edge detections on the circle. If there is a relative high amount of edge pixels in the neighborhood, the circle candidate is rejected. After filtering out static edges and using the two filtering criteria, the number of candidates should be reduced to one if only one sphere is moving.
Hough circle detection is not very accurate (see subsection 3.2.2). Hence, we refine the circle parameters (2D center point and radius) by using least-squares optimization. For each inlier edge point, we add a cost term which is the difference between the estimated radius and the distance between the edge point and the estimated center point.

Pinhole Model

A spherical camera model is not very common for standard lenses and applications. Most commonly, a pinhole model is used. As mentioned before, in the case of a pinhole model, the projection of a sphere is an ellipse. An ellipse can also be estimated based on the Hough Transform but is significantly more expensive than for circle detection because of two additional parameters. A simple but effective way to avoid the problem of ellipse estimation is to find a mapping which transforms the pinhole image to the equivalent spherical image. The sphere projection is a circle again and the efficient algorithm discussed before can be used. Creating the mapping is straightforward. Usually, a bilinear interpolation is used for interpolating pixel values.

Raw Images

If no intrinsic camera model is given, and therefore, no mapping from the input image to a spherical model can be generated, an approach that is independent of the exact shape of the projection has to be used. A reasonable assumption is that the projection of the sphere is a blob. To segment the blob, we use ArUco markers (see subsection 3.2.3) and lines that are attached to the background of the calibration target (see Figure 4.1). These markings can be efficiently detected and enable to segment the area around the blob reliably. Even if not all markings are detected because of occlusion or because they are not in the view anymore, an appropriate region of interest can be estimated and the sphere projection can be segmented.
We use edge detection with sub-pixel precision as explained in subsection 3.2.1. To filter out edges that are not on the contour of the projected sphere, we use multiple filter criteria. First, every edge point must lie inside the line marker polygon. Then, the center of mass of all the remaining edge points is calculated. The edge must be white to black from the center of mass to the outside of the blob. Finally, we assess the entirety of the edge points by determining the angle coverage of the edges relative to the center of mass. If the coverage is too low, the image is skipped.
Not only the edge position but also the connection between neighboring edge points provides information. Therefore, we want to additionally provide the tangent direction in each edge point. Theoretically, a circle fit would be a good model to approximate the local contour of the edge but, in practice, it turns out that it is unreliable. A significantly more robust approach is to fit a line through a small neighborhood around each edge point. This can be done analytically by a 2D PCA (see subsection 3.3.3).

5.2 LiDAR Sphere Detector

Common LiDARs provide point cloud data which is organized in a 2D array. Each row represents an elevation angle and each column corresponds to an azimuthal angle. A row is called a scan line. In contrast to unorganized point clouds, the neighborhood search for an organized point cloud is trivial, and therefore, very efficient. This is used in the following detection process.
As for the camera detector, we assume that the sensor setup is not moving.

We accumulate several LiDAR scans which represent the static scene. This accumulated point cloud is stored. When new point cloud data is received, we check for each point whether it is close to a static point or not. Points that are closer than a minimum distance (e.g. 10 cm) to the static scene are filtered out. For each scan line, we search for line segments with a length equal or smaller than the sphere diameter. Then, we check if line segments between neighboring scan lines overlap in their azimuthal range. Clusters over multiple scan lines are generated from these overlapping line segments. A cluster is a candidate if all its line segments are roughly centered at one azimuthal angle. We fit a sphere to the points in a cluster to make a decision whether the cluster represents a sphere well or not. The radius and position of the sphere are estimated by least-squares optimization. Since we know the real radius of our spherical target, comparing the estimated and the real radius is a strong and reliable filtering criterion. A last check is based on the ratio of outliers to inliers.
In case of a 2D LiDAR, we essentially use the same process but fit a circle to the data instead of a sphere. The radius of the circle must be equal or less than the radius of the sphere.

5.3 Radar Detector

Radar data provides less detailed geometry information about the environment compared to e.g. 3D point cloud data from LiDAR which makes it hard to segment the radar target by its shape. We use the comparably high RCS of our calibration target to filter out most irrelevant measurements. Additionally, we again assume that the target is moving, and therefore, we record the static scene and ignore data that is close to it. If another moving object with high RCS enters the scene, we reject all candidates. In the worst case, the calibration target is out of view and another object with high RCS is in the view of the radar. Then, the other object with high RCS is erroneously interpreted as the target. In a later stage of the calibration framework, such false positive detections can be filtered out by robustification techniques (see subsection 3.5.2).

6 Euclidean Calibration Framework

This chapter is linked to our publication [6].

6.1 Scope

The Euclidean calibration framework is able to estimate the sensor poses of a setup consisting of multiple cameras and range sensors. The number of sensors is not limited and the type of the range sensors is not relevant as long as the calibration target can be detected. Our spherical calibration target (see chapter 4) is used. The target is detected by the detectors introduced in chapter 5. We assume to have intrinsically precalibrated cameras so that our shape-based camera detector can be used.

6.2 Problem Formulation

The main idea of the calibration framework is to optimize the sensor poses so that the sum of squared Euclidean distances between detections of different sensors is minimized. This is visualized for a camera and a LiDAR in Figure 6.1. Mathematically, this can be formulated as following:
Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a sensor setup consisting of n sensors. The poses of the sensors are defined by the transformations $\mathcal{T} = \{\boldsymbol{T}_1, \ldots, \boldsymbol{T}_n\}$, where $\boldsymbol{T}_i \in \boldsymbol{SE}(3)$. For brevity, we omit the reference coordinate system in the subscript and simply write $\boldsymbol{T}_i$ for $\boldsymbol{T}_{\mathrm{ref},i}$ (see subsection 3.4.3). Further, we denote all sphere center detections of a sensor S_i as $\boldsymbol{X}_i$, where the detections are defined in the according sensor coordinate system. We summarize the detections of all sensors in $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$. From $\mathcal{X}$, we generate m time-synchronized pairs of observations $\mathcal{P} = \{P_1, \ldots, P_m\}$ by interpolation. We formulate our cal-

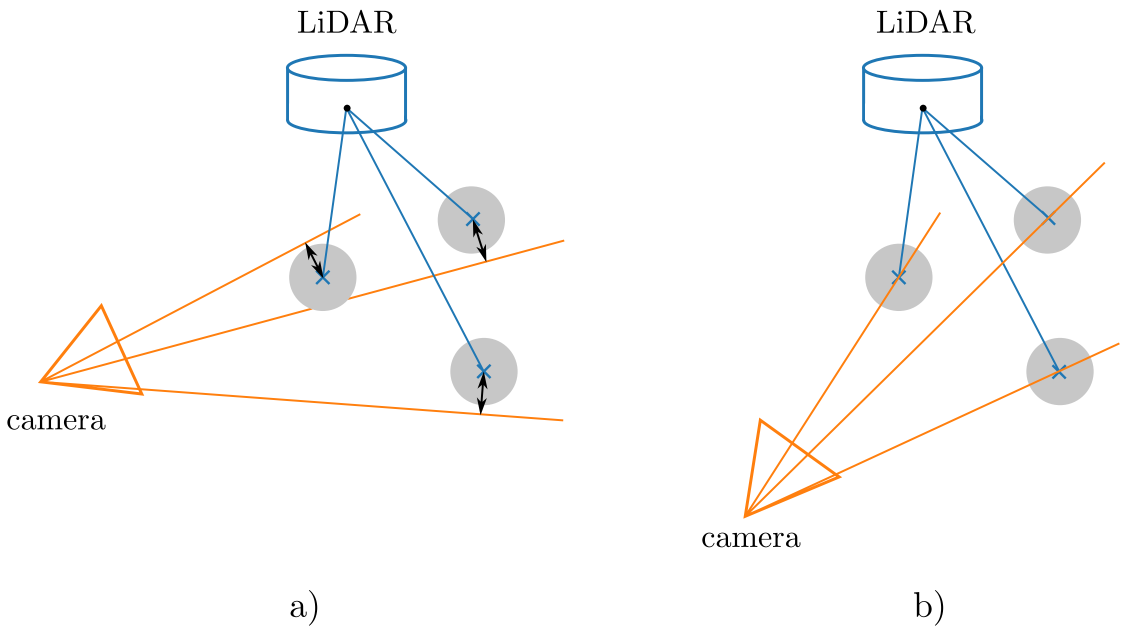


Figure 6.1: A camera and a LiDAR are calibrated. The gray circles symbolize the true position of the sphere at three times. The blue lines and crosses symbolize the 3D LiDAR detections and the orange lines represent the ray detections of the sphere center from camera. In a) the distances between the detections from the camera and LiDAR are visualized with black arrows. The distances are high because the sensor setup is uncalibrated. b) shows the sensor constellation after a successful calibration of the sensor setup. The distances between the detections are minimized.

ibration problem as the search for the transformations $\mathcal{T}$ which minimize the sum of squared distances of all sphere center observation pairs $\mathcal{P}$:

$$\arg\min_{\mathcal{T}} \sum_{i=1}^{m} \operatorname{dist}(P_i, \mathcal{T})^2 \quad . \tag{6.1}$$

The distance function $\operatorname{dist}(P_i, \mathcal{T})$ takes two arguments, first, the observation pair of which the distance shall be calculated, and second, the sensor poses to transform the observations to a common coordinate system. The types of observations decide on the distance measure that is calculated. We differentiate between ray and point observations. So, point-to-point, point-to-ray and ray-to-ray distance measures are possible. Explicit formulas for the measures are given in subsection 3.4.1.
From an image, the 3D direction to the center of the sphere can be derived with the 2D center point detection and the intrinsic camera model. The re-

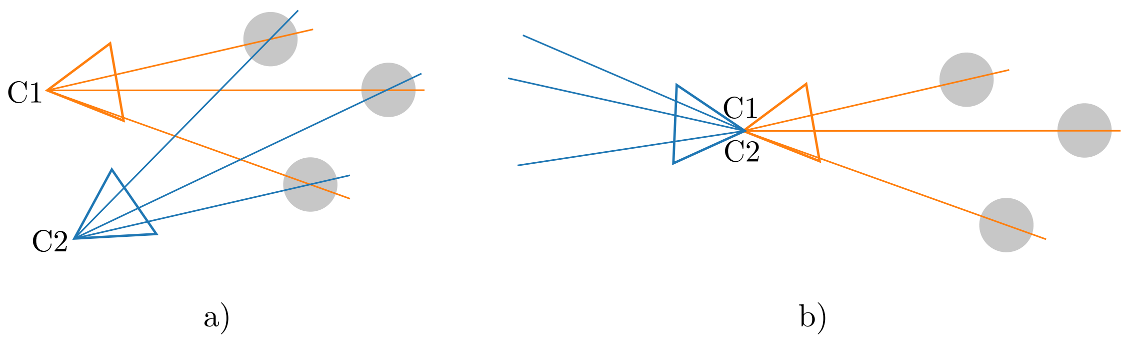


Figure 6.2: Two cameras are calibrated without range information. The optimization cost is calculated by the sum of ray-to-ray distances. a) shows the correct solution and b) shows a trivial solution for which the camera centers C1 and C2 coincide. Because all rays origin in the same point, the ray-to-ray distances are zero for the trivial solution. The blue camera can be rotated around its center to generate an infinite number of additional trivial solutions.

sulting detection is of 3D ray type. If, additionally, the radius of the sphere is given, the full 3D center position can be determined. Since the accuracy of the range estimation decreases quickly with the distance of the sphere, we usually only use the direction information. There is one special case in which the imprecise range information from camera must be taken into account. When calibrating only cameras without range information, the ray-to-ray distances are minimized for a trivial solution in which all cameras are placed at the same point (Figure 6.2). To prevent this trivial solution, at least one camera has to use range information.
LiDARs and radars provide reliable range information. Therefore, their observations are of point type.

6.3 Interpolation

As previously explained, we use time-synchronized observation pairs. Cameras can be triggered so that the images are recorded at the same time. Hence, the target detections from the cameras have the same timestamps. For other sensors, this is not always possible. E.g. we use rotating LiDARs that scan the environment continuously. Therefore, the timestamp of the LiDAR detection depends on the position of the target. So, we have to deal with asynchronous target detections.

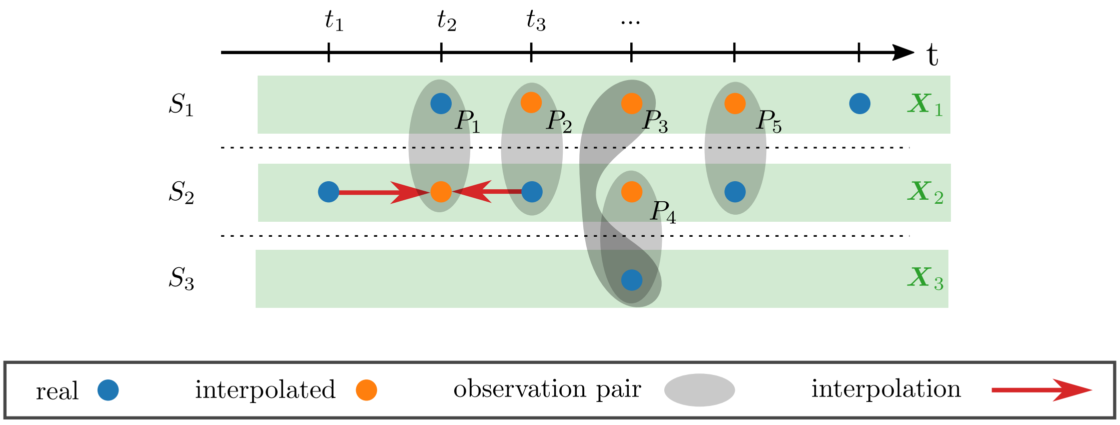


Figure 6.3: Visualization of the interpolation process for the observations $\boldsymbol{\mathcal{X}} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3\}$ of the senor setup $\boldsymbol{\mathcal{S}} = \{S_1, S_2, S_3\}$. The resulting observation pairs $\boldsymbol{\mathcal{P}} = \{P_1, P_2, P_3, P_4, P_5\}$ always consist at least of one real observation. The two red arrows exemplarily indicate the interpolation for time t_2 [6].

To generate time-synchronized observation pairs from asynchronous observations, we need to use a motion model for the target. To interpolate between ray observations, we assume that the target moves on a circular orbit around the sensors with constant velocity. Quaternion Slerp (see subsection 3.4.3) is used for this ray interpolation. For interpolation between two point observations, we assume motion on the connecting line with constant velocity. Interpolation takes place at a time t if there is an observation at t of sensor S_i and another sensor S_j provides an observation before and one after t which are close in time (e.g. 100 ms). The resulting observation pair links sensor S_i and S_j. The interpolation process is visualized in Figure 6.3

6.4 Solving the Optimization Problem

To solve the calibration problem, we use two stages. First, we find a suboptimal solution by calibrating pairs of two sensors. Then, we use this solution to initialize the calibration problem with all sensors. The main reason for this multi stage process is robustness. For a high number of outliers, solving the calibration problem with all sensors directly, often leads to unreliable results. For the first stage, we split the calibration problem in multiple subproblems. Each problem shall be robustly solvable. The resulting poses are then combined to a full transformation graph of the sensor setup which serves as an initial

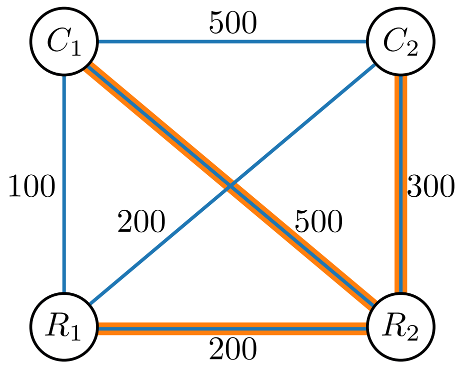


Figure 6.4: Connectivity graph of a sensor setup consisting of two cameras C_1 and C_2 and two range sensors R_1 and R_2. The number of observation pairs connecting two sensors are denoted on the corresponding edges. The maximum spanning tree which is based on the robustness measure is given by the thick orange edges.

solution for the full calibration problem.
We define a robustness measure between two sensors. Based on this measure we find a maximum spanning tree for the sensor setup. The robustness measure takes the observation types of the two sensors and the number of observation pairs that connect the two sensors into account. Point-to-point distances are the most robust, then point-to-ray distances and the least robust are ray-to-ray distances. So, a pair of two range sensors is considered to be more robust to calibrate than a camera and a range sensor which is again considered to be more robust to calibrate than a pair of two cameras. If the observation types of the sensor pairs are the same, the numbers of observations are used to prioritize. Figure 6.4 shows the connectivity graph and the maximum spanning tree of a sensor setup consisting of two cameras and two range sensors. The numbers on the connecting lines represent the count of observation pairs that link two sensors.
The robustness score does not include information about the ratio of outliers to inliers. This is because finding outliers and inliers is computationally expensive. A common way is to use a sampling based approach as explained in subsection 3.5.2, which involves solving the calibration optimization problem several times. Usually, our simple robustness measure is meaningful and leads to a good selection of subproblems, even without the information about outliers.
After the subproblems have been identified, the pairwise calibrations are solved by a robust sampling based method. As previously explained, the combined

pose graph is used to initialize the complete calibration problem which is then also solved with a robust method.

6.5 Assessment of Results

After the calibration is completed, we want to know how reliable the result is. Dividing the calibration problem in multiple subproblems allows for a simple but, in practice, very effective consistency check from which we can derive a strong statement about reliability. First, we make use of the outlier to inlier ratios. We know that our sampling based solver method can deal with around 30 % of outliers. For less outliers, the solution can be trusted with high probability. So, if all subproblems have less outliers, we reason that the initial solution should be already close to the final solution. If the final solution is close to the initial solution and the outlier ratio is less than 30 %, the solution is reliable. If high pose differences between the initialization and the final solution are detected or the outlier ratio is very high, the solution is unreliable. In this case, the calibration is solved again. If the result is reproduced, the problematic sensor pair is identified and the according data and detections can be further analyzed to find the cause of the problem. If the result is not reproducible then we had simply bad luck with the sampling based solving method. This should happen rarely and can be easily detected by running the calibration calculations several times, which takes only a few seconds.

7 Probabilistic Calibration Framework

This chapter is linked to our publication [1].

7.1 Motivation

To motivate our probabilistic calibration framework, we want to discuss the core problem of calibrating different types of sensors simultaneously.
Sensors differ vastly in their measurement principles which leads to different types of measurements and different noise characteristics. In our case, this results in different types of target detections with different detection accuracies. The detection accuracy can even differ for the same sensor, e.g. for a radar the direction accuracy is low for close objects and high for far objects. If significantly different detection characteristics are ignored, the solution can become unreliable and inaccurate.
The goal of our probabilistic calibration framework is to formulate the calibration problem in a probabilistic manner. Thereby, we want to take the detection characteristics of each sensor into account. This is the main difference to the Euclidean calibration framework (chapter 6).

7.2 Problem Formulation

Mathematically, we formulate the calibration problem as follows:
Let $\boldsymbol{\mathcal{S}} = \{S_1, ..., S_n\}$ be a sensor setup with n sensors. The pose of a sensor S_i in a reference frame $\mathcal{F}_{\text{ref}}$ is denoted by $\boldsymbol{T}_{\text{ref},i} \in \boldsymbol{SE}(3)$. Without loss of generality, we set the reference frame to the sensor frame of S_1. For brevity, we drop the reference frame in the subscript and simply write $\boldsymbol{T}_i$. All poses of

$\mathcal{S}$ are summarized in $\mathcal{T} = \{\boldsymbol{T}_1, ..., \boldsymbol{T}_n\}$. At m points in time $\boldsymbol{t} = \{t_1, ..., t_m\}$, at least two observations of the calibration target from different sensors are given. At time t_i, the position of the target in reference frame $\mathcal{F}_{\text{ref}}$ is denoted by $\boldsymbol{Z}_i$ and $\mathcal{Z} = \{\boldsymbol{Z}_1, ..., \boldsymbol{Z}_m\}$ summarizes all target positions at all times $\boldsymbol{t}$. The target position at time t_i defined in sensor coordinate system $\mathcal{F}_j$ is denoted by z_{ij} and can be calculated from $\boldsymbol{Z}_i$ and $\boldsymbol{T}_j$. A target observation at time t_i of sensor S_j in sensor frame $\mathcal{F}_j$ is written as $\boldsymbol{x}_{ij}$ and $\boldsymbol{X}_i = \{\boldsymbol{x}_{i1}, ..., \boldsymbol{x}_{in}\}$ comprises all observations of the target position at t_i. The target observations at all points in time and of all sensors are summarized in $\mathcal{X} = \{\boldsymbol{X}_1, ..., \boldsymbol{X}_m\}$. We formulate the calibration problem as a maximization problem over the joint probability density function of the unknown sensor poses $\mathcal{T}$ and target positions $\mathcal{Z}$ given all target observations $\mathcal{X}$:

$$\underset{\mathcal{T},\mathcal{Z}}{\arg\max}\ f(\mathcal{T}, \mathcal{Z} | \mathcal{X}) \ . \tag{7.1}$$

In the following, we introduce a simple but effective and practical observation model that can be used for a great variety of sensors and that leads to a simple solution of Equation 7.1:
To motivate our observation model, we want to discuss the measuring principles of common sensors. LiDARs, radars and many cameras can be modeled by using a single viewpoint from which the measurements originate. So, measurements of the same sensor are modeled by rays that have the same origin. In ray direction, cameras provide color or brightness information. LiDARs and radars provide information about the distance to the reflective surface in ray direction. The noise of a measurement can, in most cases, be well approximated by angular noise for the ray direction and noise in the additional information like brightness or range. The observation noise can be inferred by the noise of the individual measurements and the detector algorithm. Usually, the observation noise is also split in angular and range noise, if the range can be inferred. The angular noise can sometimes further be split up into azimuthal and elevation noise due to inhomogeneous angular resolution e.g. for LiDARs

with only few scan lines. To model azimuthal, elevation and range noise, we express observations in spherical coordinates:

$$\boldsymbol{x}_{ij} = \begin{cases} (r_{\boldsymbol{x}_{ij}}, \theta_{\boldsymbol{x}_{ij}}, \varphi_{\boldsymbol{x}_{ij}}) & \text{if 3D point observation} \\ (\theta_{\boldsymbol{x}_{ij}}, \varphi_{\boldsymbol{x}_{ij}}) & \text{if 3D ray observation} \\ (r_{\boldsymbol{x}_{ij}}, \varphi_{\boldsymbol{x}_{ij}}) & \text{if 2D point observation} \\ \varphi_{\boldsymbol{x}_{ij}} & \text{if 2D ray observation} \end{cases}, \tag{7.2}$$

with range r, elevation θ and azimuthal angle φ.
Mathematically, the observation model is given by the density function $f(\boldsymbol{x}_{ij}|\boldsymbol{T}_j, \boldsymbol{Z}_i)$ of the observation $\boldsymbol{x}_{ij}$ given the sensor pose $\boldsymbol{T}_j$ and the target position $\boldsymbol{Z}_i$ in the reference coordinate system. We use $f(\boldsymbol{x}_{ij}|\boldsymbol{T}_j, \boldsymbol{Z}_i) = f(\boldsymbol{x}_{ij}|z_{ij})$ as a more compact notation, where the target position z_{ij} at time t_i is given in the sensor coordinate system $\mathcal{F}_j$ of sensor S_j.
A reasonable assumption is that angles and range are independent so that we can split up the observation model:

$$f(\boldsymbol{x}_{ij}|z_{ij}) = f(r_{\boldsymbol{x}_{ij}}|r_{z_{ij}})f(\theta_{\boldsymbol{x}_{ij}}|\theta_{z_{ij}})f(\varphi_{\boldsymbol{x}_{ij}}|\varphi_{z_{ij}}) \ . \tag{7.3}$$

This will later simplify the calibration problem significantly. Note, that the subscript of a component is used to indicate the variable the component belongs to. E.g. $r_{\boldsymbol{x}_{ij}}$ is the range component of the observation $\boldsymbol{x}_{ij}$ and $r_{z_{ij}}$ is the range component of the target position z_{ij}.
At this point, we want to model each component by a distribution that describes the observation characteristic adequately:
For the range component $r_{\boldsymbol{x}_{ij}}$ of an observation, we use a one sided truncated Gaussian distribution that is created by cutting the Gaussian distribution $\mathcal{N}(r_{\boldsymbol{x}_{ij}}|r_{z_{ij}}, \sigma^2_{r_{ij}})$ at $r_{\boldsymbol{x}_{ij}} = 0$. This accounts for the fact that the range component is non-negative. If the expected observation range $\mathbb{E}(r_{\boldsymbol{x}_{ij}})$ is significantly larger than the standard deviation $\sigma_{r_{ij}}$, the distribution can be approximated with $\mathcal{N}(r_{\boldsymbol{x}_{ij}}|r_{z_{ij}}, \sigma^2_{r_{ij}})$ for $r_{\boldsymbol{x}_{ij}} \geq 0$ [JKB16]. For the angular components, we use wrapped Gaussian distributions (see [Fis95]) that are derived from $\mathcal{N}(\theta_{\boldsymbol{x}_{ij}}|\theta_{z_{ij}}, \sigma^2_{\theta_{ij}})$ and $\mathcal{N}(\varphi_{\boldsymbol{x}_{ij}}|\varphi_{z_{ij}}, \sigma^2_{\varphi_{ij}})$. Again, if the standard deviations are small, the distributions can be approximated by the corresponding normal distributions. The validity of the sensor model is shown on real data for different sensors in subsection 10.1.2.

To solve the calibration problem, we us the observation model $f(\boldsymbol{x}_{ij}|z_{ij})$ to derive a simple problem formulation for Equation 7.1 that can be solved with standard methods.
First, we use Bayes' rule (see subsubsection 3.3.1):

$$\operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} f(\mathcal{T},\mathcal{Z}|\mathcal{X}) = \operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} \frac{f(\mathcal{X}|\mathcal{T},\mathcal{Z})f(\mathcal{T},\mathcal{Z})}{f(\mathcal{X})} \quad . \tag{7.4}$$

Because we have no prior knowledge, neither on the sensor poses $\mathcal{T}$ nor on the target positions $\mathcal{Z}$, we assume $f(\mathcal{T},\mathcal{Z})$ to be independent of $\mathcal{T}$ and $\mathcal{Z}$. Furthermore, $f(\mathcal{X})$ is independent of $\mathcal{T}$ and $\mathcal{Z}$. So, these factors can be neglected in the maximization problem. The calibration problem can be reformulated as a maximum likelihood estimation:

$$\operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} f(\mathcal{T},\mathcal{Z}|\mathcal{X}) = \operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} f(\mathcal{X}|\mathcal{T},\mathcal{Z}) \quad . \tag{7.5}$$

Next, we assume that observations at different times t_i and of different sensors S_j are independent:

$$\begin{aligned}
\operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} f(\mathcal{X}|\mathcal{T},\mathcal{Z}) &= \operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} \prod_i f(\boldsymbol{X}_i|\mathcal{T},\boldsymbol{Z}_i) && (7.6)\\
&= \operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} \prod_i \prod_j f(\boldsymbol{x}_{ij}|\boldsymbol{T}_j,\boldsymbol{Z}_i) && (7.7)\\
&= \operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} \prod_i \prod_j f(\boldsymbol{x}_{ij}|z_{ij}) \quad . && (7.8)
\end{aligned}$$

In Equation 7.8, we recognize the observation model and further expand to

$$\begin{aligned}
(7.8) &= \operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} \prod_i \prod_j f(r_{\boldsymbol{x}_{ij}}|r_{z_{ij}}) f(\theta_{\boldsymbol{x}_{ij}}|\theta_{z_{ij}}) f(\varphi_{\boldsymbol{x}_{ij}}|\varphi_{z_{ij}}) && (7.9)\\
&= \operatorname*{arg\,max}_{\mathcal{T},\mathcal{Z}} \prod_i \prod_j \frac{1}{\sqrt{2\pi}\sigma_{r_{ij}}} \exp\left(-\frac{(r_{\boldsymbol{x}_{ij}} - r_{z_{ij}})^2}{2\sigma_{r_{ij}}^2}\right)\\
&\qquad \frac{1}{\sqrt{2\pi}\sigma_{\theta_{ij}}} \exp\left(-\frac{(\theta_{\boldsymbol{x}_{ij}} - \theta_{z_{ij}})^2}{2\sigma_{\theta_{ij}}^2}\right) && (7.10)
\end{aligned}$$

$$\frac{1}{\sqrt{2\pi}\sigma_{\varphi_{ij}}} \exp\left(-\frac{(\varphi_{\boldsymbol{x}_{ij}} - \varphi_{z_{ij}})^2}{2\sigma_{\varphi_{ij}}^2}\right)$$

$$= \operatorname*{arg\,min}_{\mathcal{T},\mathcal{Z}} \sum_i \sum_j \left(\frac{r_{\boldsymbol{x}_{ij}} - r_{z_{ij}}}{\sigma_{r_{ij}}}\right)^2 + \left(\frac{\theta_{\boldsymbol{x}_{ij}} - \theta_{z_{ij}}}{\sigma_{\theta_{ij}}}\right)^2 + \left(\frac{\varphi_{\boldsymbol{x}_{ij}} - \varphi_{z_{ij}}}{\sigma_{\varphi_{ij}}}\right)^2 . \tag{7.11}$$

We use the strictly increasing logarithm to derive Equation 7.11. This formulation is a least-squares problem that can be efficiently solved with the techniques described in subsection 3.5.2. The residual vector consists of the differences in components of the observations $\boldsymbol{x}_{ij}$ and the according target positions z_{ij} that are normed with the standard deviation for each component at the according time. In Equation 7.9 we assume 3D point observations defined by two angles and the range. If other observation types are used (see Equation 7.2), the according non-existent components are left out in the least-squares problem (Equation 7.11). Note that at least one sensor should provide range information so that scale can be inferred.

7.3 Determining the Observation Noise

At this point, we derived a simple least-squares problem from our probabilistic formulation in Equation 7.1 by using our universal observation model. For solving the least-squares problem, we need to know the observation standard deviations for each component at the according times. As previously explained, the observation noise results from the measurement noise which is propagated through the detection algorithm. Calculating the observation noise based on the detection algorithm and the measurement noise is usually difficult. A more practical alternative is to experimentally determine the observation noise. Therefor, the position of the target relative to the sensor must be known. This usually involves a very elaborate procedure and expensive hardware. A more practical approach is to use simulation. Compared to the effort needed for real experiments, it is often easier to create a suitable simulation.
The simplest approach to derive the observation noise is intelligent guessing. Based on the measurement noise of the sensors, it is easy to estimate the

components with highest and lowest noise. Of course, this only allows for a rough estimate of the observation noise but the question is, how accurate does the estimate have to be to reach good results? Actually this is the most important question for the practical use of this calibration method.
In our evaluation (see chapter 10), we analyze the influence of using information about observation noise in the calibration problem and show the sensitivity of the calibration quality with regard to errors in the noise estimates. Further, we give an example of how to experimentally derive observation noise on real data and in simulation. Finally, we compare our probabilistic approach to the Euclidean calibration framework that ignores noise characteristics.

8 Joint Calibration Framework

The main idea of this chapter is linked to our publication [5]. The definition of the residuals in section 8.4 is fundamentally different and is an advancement of [5].

8.1 Motivation

The probabilistic calibration framework introduced in the previous chapter can outperform the Euclidean calibration framework significantly by using information about the target observation noise (see section 10.3). The universal observation model is very simple but effective and can be used for a great variety of sensors. But, because it is so unspecific for a sensor, it is often inconvenient to model all characteristics in detail. E.g. the angle accuracy of an observation from a camera depends on the position of the sphere's projection in the image. Theoretically, this can be modeled by expressing the angle noise as a function of the projection's position. However, deriving this function is hard. Therefore, the universal observation model is only practical if the observation characteristics are considered constant for a sensor. But even then, the estimation of the observation noise is not trivial. As explained before, an accurate estimate involves elaborate experiments on simulated or real data. Estimating the target observation noise is in practice significantly harder than finding the noise of the raw measurements because this information is usually given in the datasheet of a sensor. So, ideally, the user only has to provide the measurement noise information and does not have to estimate the target observation noise. Therefore, we want the raw measurements as input to the calibration problem instead of sphere center observations. Besides the benefit that no observation noise has to be estimated, working on raw measurements also allows for increased calibration quality because more information is used compared to when working with sphere center observations.

One limitation of the base and probabilistic calibration framework is that cameras have to be intrinsically calibrated before extrinsic parameters can be estimated. So, another method has to be used to calibrate each camera intrinsically which is an elaborate preparation procedure. It would be more convenient if camera intrinsics are estimated while extrinsics are calibrated. This way, only a single dataset with a single calibration target would be needed. Additionally, calibrating intrinsic and extrinsic parameters in a joint calibration problem can, theoretically, lead to improved estimates of intrinsic parameters by complementary information from different sensors. E.g. the distance of the target cannot be accurately estimated in far distance from camera image data. The range information of a LiDAR can help estimating the target distance more accurately, and therefore, can improve camera intrinsics.

We summarize the goals for the joint calibration framework as follows: Intrinsic and extrinsic parameters are calibrated in a joint calibration problem. No other method is needed to calibrate the intrinsics of the cameras beforehand. Raw measurements are used instead of sphere center observations. As a result, the user has to provide information about the measurement noise instead of the observation noise which is more convenient.

8.2 Input Data

Because raw measurements instead of sphere detections shall be used, the sphere detectors used in the Euclidean and probabilistic calibration framework are replaced by segmentation stages that only isolate relevant measurements but do not fit a sphere to the data. For LiDAR, points that hit the surface of the sphere are segmented. The filter criteria described in section 5.3 are used. Radar data is used as before because the radar reflector is positioned in the center of the sphere so that the radar measurements are the same as the sphere center observations. The segmentation stage for camera images extracts the edge contours of the sphere projections which serve as input data. Details about the segmentation stage on camera images are given in section 5.1.

8.3 Problem Formulation

Let's define the joint calibration problem mathematically:
Given is a sensor setup $\boldsymbol{\mathcal{S}} = \{S_1, ..., S_l\}$. In addition to its pose $\boldsymbol{T}_i$, we describe a sensor S_i by a set of intrinsic parameters $\boldsymbol{I}_i$ and summarize all parameters of S_i in $\boldsymbol{\theta}_i = \{\boldsymbol{T}_i, \boldsymbol{I}_i\}$. The parameters of the sensor setup $\boldsymbol{\mathcal{S}}$ are given by $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_l\}$. A sensor S_i at time t_j provides $n_{i,j}$ measurements on the target that are summarized in $\boldsymbol{X}_{i,j} = \{\boldsymbol{x}_{i,j,1}, ..., \boldsymbol{x}_{i,j,n_{i,j}}\}$. We denote measurements of all sensors at time t_j as $\boldsymbol{X}_j = \{\boldsymbol{X}_{1,j}, ..., \boldsymbol{X}_{l,j}\}$ and further summarize all measurements from all sensors at all times $t_1, ..., t_m$ as $\boldsymbol{\mathcal{X}} = \{\boldsymbol{X}_1, ..., \boldsymbol{X}_m\}$. The target position at time t_j is denoted as $\boldsymbol{Z}_j$ and all target positions at all times are given by $\boldsymbol{\mathcal{Z}} = \{\boldsymbol{Z}_1, ..., \boldsymbol{Z}_m\}$.
As in chapter 7, we define the calibration problem in a probabilistic manner:

$$\underset{\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}}{\arg\max} \; f(\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}} | \boldsymbol{\mathcal{X}}) \;, \tag{8.1}$$

where $f(\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}} | \boldsymbol{\mathcal{X}})$ is the joint probability density function of the sensor parameters $\boldsymbol{\theta}$ and the target positions $\boldsymbol{\mathcal{Z}}$ given the measurements $\boldsymbol{\mathcal{X}}$.
To solve Equation 8.1, we need to find an explicit expression for $f(\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}} | \boldsymbol{\mathcal{X}})$. We assume to have a measurement model $f(\boldsymbol{x}_{i,j,k} | \boldsymbol{\theta}_i, \boldsymbol{Z}_j)$ for each measurement $\boldsymbol{x}_{i,j,k}$. We use Bayes' rule to reformulate the calibration problem as a maximum likelihood estimator as described in chapter 7

$$\underset{\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}}{\arg\max} \; f(\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}} | \boldsymbol{\mathcal{X}}) = \underset{\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}}{\arg\max} \; f(\boldsymbol{\mathcal{X}} | \boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}) \;, \tag{8.2}$$

which is based on the assumption that no prior knowledge on the parameters $\boldsymbol{\theta}$ and the target positions $\boldsymbol{\mathcal{Z}}$ is given. To incorporate the measurement model for each single measurement, we assume independence of the measurements that leads to

$$\underset{\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}}{\arg\max} \; f(\boldsymbol{\mathcal{X}} | \boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}) = \underset{\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}}{\arg\max} \prod_{i=1}^{l} \prod_{j=1}^{m} \prod_{k=1}^{n_{i,j}} f(\boldsymbol{x}_{i,j,k} | \boldsymbol{\theta}_i, \boldsymbol{Z}_j) \;. \tag{8.3}$$

For universal consideration, we introduce residuals $r_{i,j,k}(\boldsymbol{x}_{i,j,k}, \boldsymbol{\theta}_i, \boldsymbol{Z}_j)$ that are used to define the measurement model $f(\boldsymbol{x}_{i,j,k} | \boldsymbol{\theta}_i, \boldsymbol{Z}_j)$. We will show in section 8.4 that the residuals can be assumed to be zero-mean Gaussian which leads to the measurement model

$$f(\boldsymbol{x}_{i,j,k}|\boldsymbol{\theta}_i, \boldsymbol{Z}_j) = \mathcal{N}\left(r_{i,j,k}\middle|0, \sigma^2_{r_{i,j,k}}\right) \tag{8.4}$$

and results in a simple least-squares minimization problem

$$\underset{\boldsymbol{\theta},\boldsymbol{\mathcal{Z}}}{\arg\max}\ f(\boldsymbol{\mathcal{X}}|\boldsymbol{\theta}, \boldsymbol{\mathcal{Z}}) = \underset{\boldsymbol{\theta},\boldsymbol{\mathcal{Z}}}{\arg\min} \sum_{i=1}^{l} \sum_{j=1}^{m} \sum_{k=1}^{n_{i,j}} \left(\frac{r_{i,j,k}}{\sigma_{r_{i,j,k}}}\right)^2 . \tag{8.5}$$

This problem formulation differs from the one introduced in chapter 7 in two points. First, for one point in time t_j, multiple measurements instead of one observation are used. Second, the residual is not limited to differences in spherical coordinates. The residuals can be more specific to the sensor, and thereby, can be more suitable to model the characteristics. This becomes obvious when noting that measurements $\boldsymbol{X}_{i,j} = \{\boldsymbol{x}_{i,j,1}, ..., \boldsymbol{x}_{i,j,n_{i,j}}\}$ of the same sensor S_i at the same time t_j can be modeled to have different standard deviations $\sigma_{i,j,1}, ..., \sigma_{i,j,n_{i,j}}$. The method in chapter 7 cannot model noise characteristics at this low level.

8.4 Residual Definitions

In this section, we introduce the residuals for radars, LiDARs and cameras.

8.4.1 Residual for Radar

In the case of radar, the segmented measurements are identical to the sphere center observations which we used in the probabilistic calibration framework. In the datasheet of radars, the measurement accuracy of the azimuthal angle and of the range measurement are given. The measurement noise in angular component and range is assumed to be zero-mean Gaussian. Therefore, we define an angular and range residual which represent the difference between the estimated target position $z_{i,j}$ and the measurement $\boldsymbol{x}_{i,j}$ in the radar scan plane.

8.4.2 Residual for LiDAR

Let's consider a 3D point measurement from LiDAR that we denote by $\boldsymbol{x}^l$. Usually, LiDARs provide high accuracy in direction but show significant noise in their range measurement. We model the range $\|\boldsymbol{x}^l\|$ by a Gaussian distribution with a standard deviation σ_{range} that can be taken from the datasheet of the sensor. Besides the random range error, there is also a systematic range error that depends on the type of surface and its relative pose to the sensor. The data is preprocessed in the sensor to reduce systematic range errors but the quality varies by a large extend depending on the LiDAR [LDLP19]. Modern LiDARs show low sensitivity in range accuracy to the distance of the reflective surface and to the reflectivity of the surface, whereas the incident angle can influence the range accuracy significantly. For many LiDARs, at incident angles larger than $70\,^\circ$, the systematic range error is around one standard deviation of the random range error. Since we know the shape of our calibration target and estimate its position, we can also derive the incident angle for each measurement. We propose two different approaches to prevent significant range errors due to high incident angles. A simple approach is to filter out measurements with high incident angles in a preprocessing step. This can be achieved by fitting a sphere to the point cloud. Actually, this is already done as a filtering step to segment measurements that hit the sphere. Hence, filtering out measurements with high incident angles comes for free. Another approach is to model the systematic range error as a function of the incident angle γ_{in}, which is measured relative to the surface normal. E.g. a polynomial function

$$b(\gamma_{\text{in}}) = p_0\gamma_{\text{in}}^2 + p_1\gamma_{\text{in}}^4 \tag{8.6}$$

can be used to model the systematic range error $b(\gamma_{\text{in}})$. The unknown parameters p_0 and p_1 can be incorporated as intrinsic parameters of the LiDAR and are estimated in the calibration process. The corrected measurement can then be calculated by

$$\boldsymbol{x}^{l,\text{cor}}(\gamma_{\text{in}}) = (\|\boldsymbol{x}^l\| - b(\gamma_{\text{in}}))\frac{\boldsymbol{x}^l}{\|\boldsymbol{x}^l\|} \quad . \tag{8.7}$$

We define the residual for LiDAR based on the range measurement. With the assumed target positions $\mathcal{Z}$ we can calculate the expected intersection point

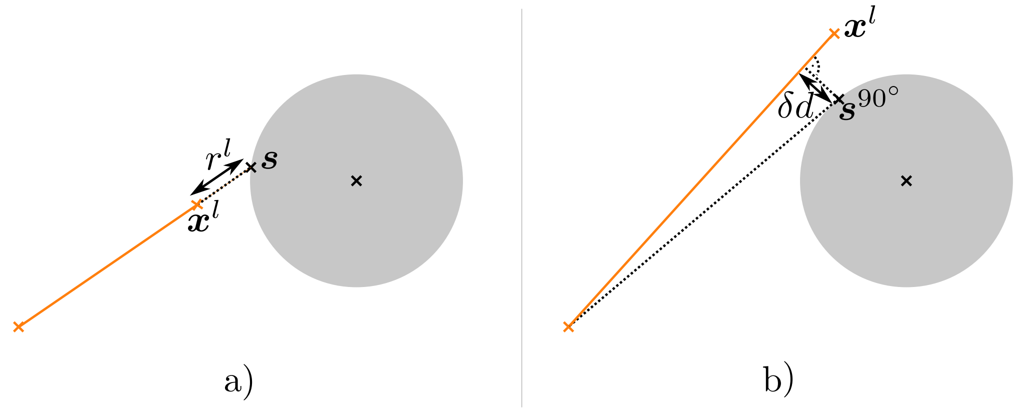


Figure 8.1: Visualization of the LiDAR residual.
a) Residual for a LiDAR point measurement $\boldsymbol{x}^l$ in the case that an intersection point $\boldsymbol{s}$ exists.
b) For the case that no intersection point between sphere and LiDAR measurement ray exists, the residual is split into two parts: first, the range difference between the measured point $\boldsymbol{x}^l$ and the point $\boldsymbol{s}^{90°}$, and second, the distance between the measurement ray and $\boldsymbol{s}^{90°}$.

$\boldsymbol{s}$ of the sphere and the ray. The difference between the range measurement and the range of the intersection point $\boldsymbol{s}$ defines the residual (see Figure 8.1). However, we have to consider a special case that can happen especially in early iterations of the calibration. If there is no intersection between the measured ray and the estimated sphere, the residual has to be defined differently. Each measurement that is considered in the calibration problem is segmented in the preprocessing stage and is therefore assumed to be on the sphere. The fact that there is no intersection is caused by large errors in the sensor poses and/or the target positions. Hence, we want to introduce high costs for the special case that measurements do not intersect with the estimated sphere. To formulate a smooth cost transition between the special case of no intersection and the normal case of intersection, we use the intersection point $\boldsymbol{s}^{90°}$ with an incident angle of 90 °. The costs consist of two parts, first, the range difference between the measurement and $\boldsymbol{s}^{90°}$, and second, the distance δd of the measurement ray and $\boldsymbol{s}^{90°}$ which penalizes the ray missing the sphere (see Figure 8.1). We summarize the definition of the LiDAR residual as following:

$$r^l = \begin{cases} \|\boldsymbol{x}^l\| - \|\boldsymbol{s}\| & \text{if } \exists \boldsymbol{s} \\ \big|\|\boldsymbol{x}^l\| - \|\boldsymbol{s}^{90°}\|\big| + \delta d & \text{else} \end{cases} . \tag{8.8}$$

Note the continuous transition between the two cases which is important for solving the optimization problem. If systematic range errors are corrected then we replace $\boldsymbol{x}^l$ by $\boldsymbol{x}^{l,\mathrm{cor}}$.
The calibration problem in Equation 8.1 assumes residuals to be zero-mean Gaussian. Let's analyze the distribution of the LiDAR residual: First, we notice that for given parameters $\boldsymbol{\theta}$ and given target positions $\boldsymbol{\mathcal{Z}}$ the intersection point $\boldsymbol{s}$ is fixed. So, if an intersection point exists, the residual r^l is distributed like the range measurement $\|\boldsymbol{x}^l\|$ which is modeled as zero-mean Gaussian with standard deviation σ_{range}. If no intersection point exists, the residual r^l is also distributed like the range measurement $\|\boldsymbol{x}^l\|$ because $\|\boldsymbol{s}^{90^\circ}\|$ and δd are not influenced by a random error.

8.4.3 Residual for Camera

Finally, we introduce residuals for camera measurements. The segmentation stage provides edge positions on the contour of the projected sphere with sub-pixel precision. There are two possible ways to define residuals, either in 2D image space or in 3D space. The suitability and practicability depends on how the camera model is defined. As explained in subsection 3.1.1, we differentiate between forward and backward models. A forward model explicitly defines how a 3D point is projected onto the image. In this case, it is more suitable to define the residual in the image space since the estimated target sphere can be easily projected onto the image. In subsection 3.1.1, we introduce a backward camera model that can be used for a great variety of different cameras. This backward model defines an explicit mapping from a 2D image point to a 3D ray in the camera coordinate system. The other way round, from 3D direction to 2D image point, is not easy to calculate with this model. Hence, suitable residuals are defined in 3D space.

For each 2D edge point $\boldsymbol{x}^c$, we calculate the corresponding viewing ray $\mathcal{P}_b(\boldsymbol{x}^c)$. Additionally, we estimate the tangent in $\boldsymbol{x}^c$ by fitting a line through a small neighborhood $\boldsymbol{x}^c_{n1}, ..., \boldsymbol{x}^c_{nu}$ of u edge points. A circle fit would seem to be more appropriate but, in practice, it turns out to be less reliable than a line fit. We use PCA (see subsection 3.3.3) to calculate the angle $\varphi(\boldsymbol{x}^c_{n1}, ..., \boldsymbol{x}^c_{nu})$ of the tangent (see Figure 8.2). This is very efficient since 2D eigenvectors can be calculated analytically. The angle $\varphi(\boldsymbol{x}^c_{n1}, ..., \boldsymbol{x}^c_{nu})$ also includes the information on which side of the tangent the sphere lies. To transform the edge direction

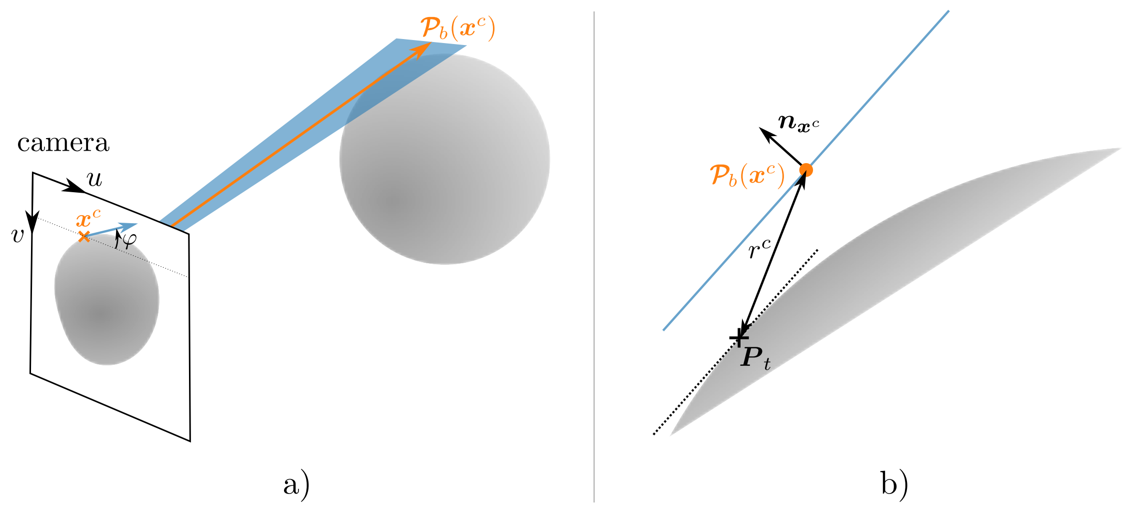

$\mathcal{P}_b(\boldsymbol{x}^c)$
camera
u
v
$\boldsymbol{x}^c$
φ
a)
$\boldsymbol{n}_{\boldsymbol{x}^c}$
$\mathcal{P}_b(\boldsymbol{x}^c)$
r^c
$\boldsymbol{P}_t$
b)

Figure 8.2: Visualization to explain the camera residual. a) The information of the 2D edge point $\boldsymbol{x}^c$ with according tangent direction is transformed to 3D space in the form of the 3D viewing ray $\mathcal{P}_b(\boldsymbol{x}^c)$ and the according tangent plane. b) The residual r^c is defined by using the viewing ray $\mathcal{P}_b(\boldsymbol{x}^c)$ (orange point symbolizes that the ray is directed into the plane of drawing) and the tangent point $\boldsymbol{P}_t$ that is the closest point on the sphere surface to the tangent plane (appears as the blue line in this 2D cut).

into 3D space, we locally define a 3D tangent plane (see Figure 8.2). The ray $\mathcal{P}_b(\boldsymbol{x}^c)$ is on the plane and its projection to the image has the same 2D tangent in $\boldsymbol{x}^c$. The normal direction can be calculated by

$$\boldsymbol{n}_{\boldsymbol{x}^c} = \mathcal{P}_b(\boldsymbol{x}^c) \times \nabla^{\parallel}\mathcal{P}_b(\boldsymbol{x}^c) \ , \tag{8.9}$$

where $\nabla^{\parallel}$ denotes the derivative in direction parallel to the 2D tangent in $\boldsymbol{x}^c$. We calculate the closest point $\boldsymbol{P}_t$ to the tangent plane that lies on the estimated sphere surface. The residual r^c is defined by the point-to-ray distance $d(\cdot,\cdot)$ (see subsection 3.4.1) between the point $\boldsymbol{P}_t$ and the viewing ray $\mathcal{P}_b(\boldsymbol{x}^c)$:

$$r^c = d(\boldsymbol{P}_t, \mathcal{P}_b(\boldsymbol{x}^c)) \ . \tag{8.10}$$

But why do we introduce $\boldsymbol{P}_t$ to define the residual? An initial idea could be to simply define the residual as the distance between viewing ray and sphere surface. This definition has a major problem. There is a trivial solution for which all estimated target spheres are positioned very close to the camera so that the origin of the viewing rays are very close to the sphere surfaces. To prevent this, we use the additional information about the direction of the tangent. If the orientation of the tangent plane is consistent with the center point of the estimated sphere, our residual represents the minimum distance between viewing ray and sphere.

To use the residual, we need to analyze its distribution. First, we notice that the residual depends on the camera model $\mathcal{P}_b$. Hence, the distribution of the residual also depends on the camera model. In practice, the noise is small so that the residual can be locally approximated well by a linear function. This allows us to use the concept of error propagation discussed in subsection 3.3.4. With the common assumption that edge detections are zero-mean Gaussian orthogonal to the edge direction, we can propagate the Gaussian through the calculation of the residual and derive a locally valid Gaussian approximation for the residual distribution: We denote the standard deviation of the edge detection by σ_{pix}. From this, we calculate the variance σ_φ^2 of the tangent angle $\varphi(\boldsymbol{x}_{n1}^c, ..., \boldsymbol{x}_{nu}^c)$. We use the error propagation formula of Equation 3.31 to infer

$$\sigma_\varphi^2 = \nabla^\perp \varphi^T(\boldsymbol{x}_{n1}^c, ..., \boldsymbol{x}_{nu}^c) \begin{pmatrix} \sigma_{\text{pix}}^2 & 0 & \dots & 0 \\ 0 & \sigma_{\text{pix}}^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sigma_{\text{pix}}^2 \end{pmatrix} \nabla^\perp \varphi(\boldsymbol{x}_{n1}^c, ..., \boldsymbol{x}_{nu}^c) \quad (8.11)$$

$$= \|\nabla^\perp \varphi(\boldsymbol{x}_{n1}^c, ..., \boldsymbol{x}_{nu}^c)\|^2 \sigma_{\text{pix}}^2 \ , \quad (8.12)$$

where $\nabla^\perp$ is the derivative in direction orthogonal to the tangent in each neighborhood point $\boldsymbol{x}_{n1}, ..., \boldsymbol{x}_{nu}$. Since we have an analytical expression for the tangent angle $\varphi(\boldsymbol{x}_{n1}^c, ..., \boldsymbol{x}_{nu}^c)$, the variance σ_φ^2 is also calculated analytically (details are given in section A.2). The residual $r^c(\boldsymbol{x}^c, \varphi)$ depends on the edge point $\boldsymbol{x}^c$ and tangent direction φ and is influenced by the tangent angle noise and the noise of the edge point orthogonal to the tangent. With the concept of error propagation, we conclude that the variance of the residual r^c can be calculated by

$$\sigma_{r^c}^2 = \|\nabla_\varphi r^c(\boldsymbol{x}^c, \varphi)\|^2 \sigma_\varphi^2 + \|\nabla_{\boldsymbol{x}^c}^\perp r^c(\boldsymbol{x}^c, \varphi)\|^2 \sigma_{\text{pix}}^2 \quad (8.13)$$

$$= \left[\|\nabla_\varphi r^c(\boldsymbol{x}^c, \varphi)\|^2 \|\nabla^\perp \varphi(\boldsymbol{x}_{n1}^c, ..., \boldsymbol{x}_{nu}^c)\|^2 + \|\nabla_{\boldsymbol{x}^c}^\perp r^c(\boldsymbol{x}^c, \varphi)\|^2\right] \sigma_{\text{pix}}^2 \ . \quad (8.14)$$

We used the assumption that $\boldsymbol{x}^c$ and φ are uncorrelated (see Equation 3.33).

Besides the assumption of Gaussian distributed residuals, we also use the assumption that residuals are independent. This assumption is introduced to derive a very simple formulation of the calibration problem. The assumption is fulfilled for radar and also LiDAR but the camera residual seems to be problematic because we use the neighborhood around an edge point to calculate the tangent direction. If an edge point is used in multiple neighborhoods, the according residuals are not independent anymore. To prevent this, we do not use edge points twice. So, the edge contour is sampled in a way that neighborhoods do not overlap.

8.5 Solving the Optimization Problem

At this point, we know how to calculate the residuals and the according standard deviations and could, theoretically, solve the optimization problem in Equation 8.5. Solving this problem with an initialization procedure that solves several subproblems has many advantages:
First, the subproblems have less local minima than the complete optimization problem. By initializing the complete problem with the solutions of the subproblems, the risk of getting stuck in a local minimum is reduced. Second, computing an iteration of the iterative solving method (see subsection 3.5.2) is expensive for this comparably large calibration problem. An efficient initialization routine that provides a fast but rough estimate for the parameters can reduce the number of iterations significantly, and therefore, speeds up the calibration procedure. Third, as mentioned in chapter 6, comparing the results of subproblems to the final solution of the complete problem provides a tool to detect failed calibrations and further identify the reason for an inaccurate calibration.

We use the following procedure to solve the calibration problem:
In the first stage, we pick a single camera. We formulate the calibration problem (Equation 8.5) for this single camera to get an initial estimate of its intrinsic parameters.
In the next step, the calibration problem is extended by sphere center observations of all 3D range sensors. The center observations are time-synchronized

to the camera images by the interpolation method explained in chapter 6. The optimization problem is initialized with the previously calculated intrinsics of the camera. The result provides improved intrinsics of the single camera and the poses of all 3D range sensors relative to the camera. This step is motivated by the fact that 3D range sensors are calibrated very robustly and efficiently with the sphere center observations. These observations provide strong and reliable constraints on the target positions. The accurately estimated target positions improve the estimation of the camera intrinsics. These steps are repeated for every camera.
The next stage incorporates all sensors at once. This way, the cameras can profit from each other. Again, sphere center observations are used for the range sensors. This stage refines the initial solution in all intrinsic and extrinsic parameters. The computation is very quick because of the good initialization.
Finally, we drop the center observations for the range sensors and calibrate with measurements only. No time-synchronization is used anymore but a spline is used to continuously model the target position (see subsection 3.4.2). The spline is initialized by a least-squares optimization problem which minimizes the distance of the spline to the previously estimated target positions. At this point, we solve the calibration problem for all intrinsic parameters, the sensor poses and the target trajectory. This represents our most refined solution and is then compared to the results of the subproblems for a consistency check (comparable to chapter 6).

In the beginning of this chapter, we argue that using the single measurements in the calibration problem can improve calibration accuracy compared to when using target observations. The downside is that the problem is significantly more expensive to calculate because, typically, the number of residuals is larger by a factor of $10^2 - 10^3$. Additionally, the standard deviations for camera residuals are recalculated for each iteration of the optimization problem due to the changing intrinsics. To reduce the computational demand, we can fix the standard deviations for multiple iterations. In practice, this speeds up the optimization significantly, whereas the calibration accuracy does not noticeably decrease.

9 External Keypoint Calibration

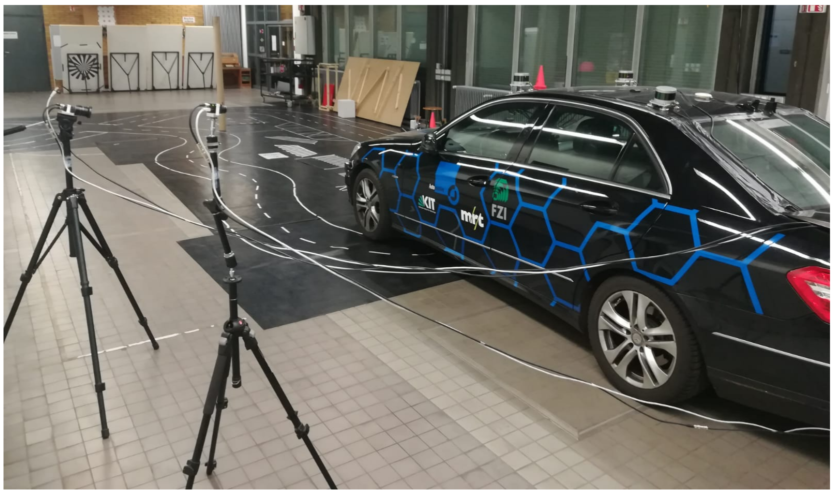
In the previous chapters, we introduced methods for calibrating a setup consisting of multiple sensors. The sensor poses are estimated relative to a reference coordinate system which could be one of the sensor coordinate systems or defined in any point on the platform on which the sensor setup is mounted. E.g. for cars, it is important to know how the sensors are positioned relative to the rear axle in order to fuse sensor data with wheel encoder and steering angle readings. Another scenario would be to fuse data from cameras and LiDARs with GNSS data for localization. The previously discussed calibration methods do not include the calibration of GNSS receivers. The position of the receiver antenna relative to the other sensors must be estimated differently. In the following, we use the term *external keypoint calibration* for the process of calibrating the main sensor setup relative to keypoints on the sensor platform such as the rear axle or GNSS antennas. The following details are linked to our publication [4].

To include external keypoints into the calibration framework, we use additional sensors that can observe the keypoints and reconstruct their 3D positions. A useful sensor for this task is an RGB-D sensor which provides visual and range information to easily localize salient keypoints in 3D. Alternatively, a stereo camera pair can be used. The 3D keypoint position can be reconstructed based on triangulation. Cameras usually offer more control than an RGB-D sensor which might have inaccurately known parameters e.g. the distance between pattern projector and camera. Therefore, cameras are preferred for applications with a high demand for accuracy.
The external sensors are calibrated to the main sensor setup with one of the methods discussed in previous chapters. Then, the keypoints can be detected in the data of the external sensors and the 3D positions can be reconstructed relative to the other sensors.

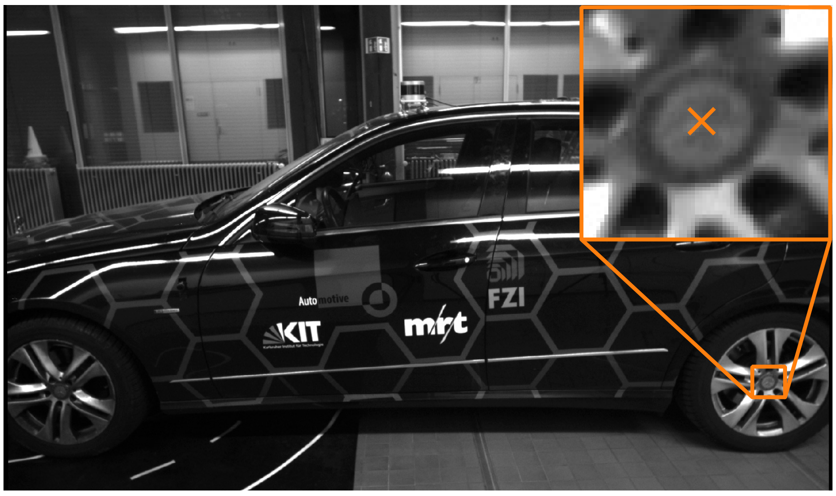
For autonomous driving, the method can be used for linking the sensor setup to the rear axle which is usually the reference for the vehicle and collision model of the car. The rear axle reference is defined by a coordinate system which has its origin centered between the two rear wheel centers, its y-axis aligns with the connecting line of the two wheel centers and its z-axis is orthogonal to the ground surface and points up. In the following, we introduce the full procedure of rear axle calibration with two external cameras. The procedure can be split up in seven steps:

1. Place the external cameras at one side of the car so that at least one wheel can be observed by both cameras. The camera poses are given unique identification numbers.
2. Record a dataset with the calibration target. The target must be observable for the external cameras and the main sensor setup.
3. Mark the center wheel position in one image for each of the external cameras (see Figure 9.1).
4. The steps 1 - 3 are repeated with different camera positions in order to observe all four wheel centers.
5. With the collected data in step 2, we calibrate the main sensor setup and the external cameras for each of their pose.
6. With the estimated poses of the external cameras and the wheel centers marked in the according images, the 3D positions of the wheel centers are reconstructed by triangulation.
7. The rear axle coordinate system relative to the main sensor setup is inferred from the four wheel center positions. The origin of the rear axle coordinate system and the y-axis are determined by the two wheels at the rear. The x-axis and z-axis are inferred by estimating the ground plane from the four wheels.

a)



b)



c)

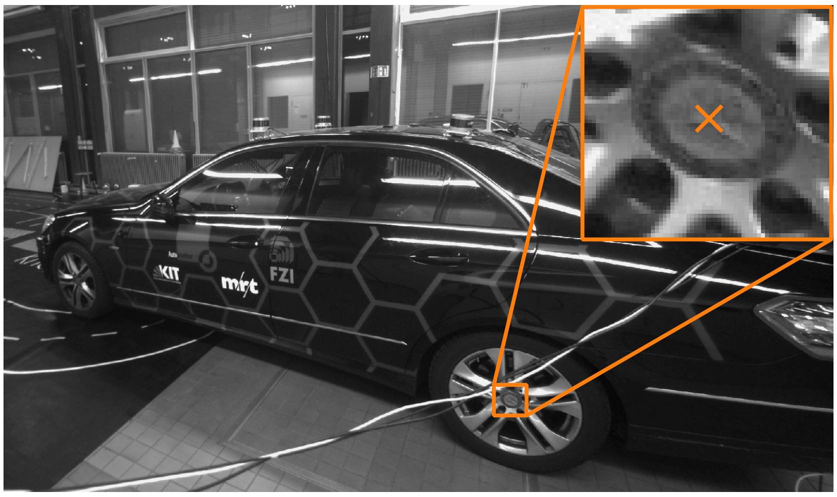


Figure 9.1: Two gray-scale cameras are positioned in front of the left wheel at the rear of our experimental vehicle. a) shows the complete setup. b) and c) show the camera images of the two cameras with marked wheel center points.

10 Evaluation

In this chapter, we evaluate our previously introduced calibration approaches. First, the sphere detectors are analyzed (section 10.1). Then, the three different calibration frameworks are evaluated and compared to each other (section 10.2 - 10.4). Finally, we evaluate the external keypoint estimation method based on the application of rear axle calibration (see section 10.5). This chapter is linked to our publications [1, 4–6].

10.1 Sphere Detection

Our sphere detectors are used to generate sphere center observations for cameras and LiDARs. In the following, we evaluate the accuracy of the detectors, identify error sources and analyze the error distributions.

10.1.1 Accuracy and Error Sources

We want to quantify the accuracy of the sphere detectors under different conditions and identify error sources that noticeably decrease the detection quality. For this to analyze, we need full control over the environment in which the calibration takes place. Therefore, we create our own simulation environment to produce raw images and point cloud data under controlled conditions. Different scenes which are derived from real data are used to make the simulation realistic (see Figure 10.1 for an example scene). In these scenes, we simulate the spherical calibration target. For cameras, the 3D sphere is rendered with different lighting, simulated defocus and pixel noise. Range noise is simulated to generate realistic measurements for LiDARs. We generate multiple datasets from simulation. The sphere moves on a trajectory around the sensor setup in a distance of $2 - 8\,\mathrm{m}$. To compare the accuracy of the detectors, we calculate

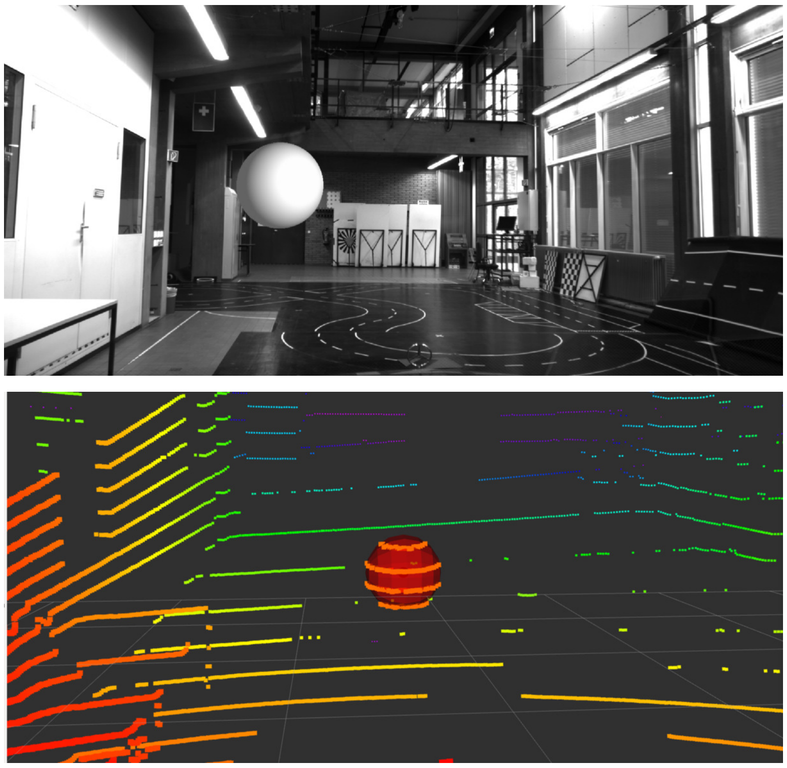


Figure 10.1: Simulation environment for evaluation. Real sensor data is used to build up a realistic scene. The top image is an example for a rendered camera image and the bottom image shows a 3D point cloud for the same scene at the same time [6].

point-to-point distances between estimated and true sphere centers.

First, we evaluate the camera sphere detector. We simulate a camera with a resolution of 2000×974 and a focal length of 1222 px. The ideal conditions for a camera are homogeneous lighting, no defocus and no pixel noise. Additionally, the background has a high contrast to the sphere and has no texture. We use a white sphere in an all-black environment. Even for these ideal conditions, the Euclidean distances of the detections to the ground truth center positions are not always zero (see Figure 10.2). This is due to discretization errors which depend on the exact position and size of the projection in the image. Next, we add a realistic background and lighting. The contrast changes around the projected sphere contour which leads to a median error that is almost twice as high as for ideal conditions. Then, we additionally simulate defocus and pixel noise. We use a Gaussian filter that is applied to the image to emulate defocus. By comparing to real data, we found that a standard deviation

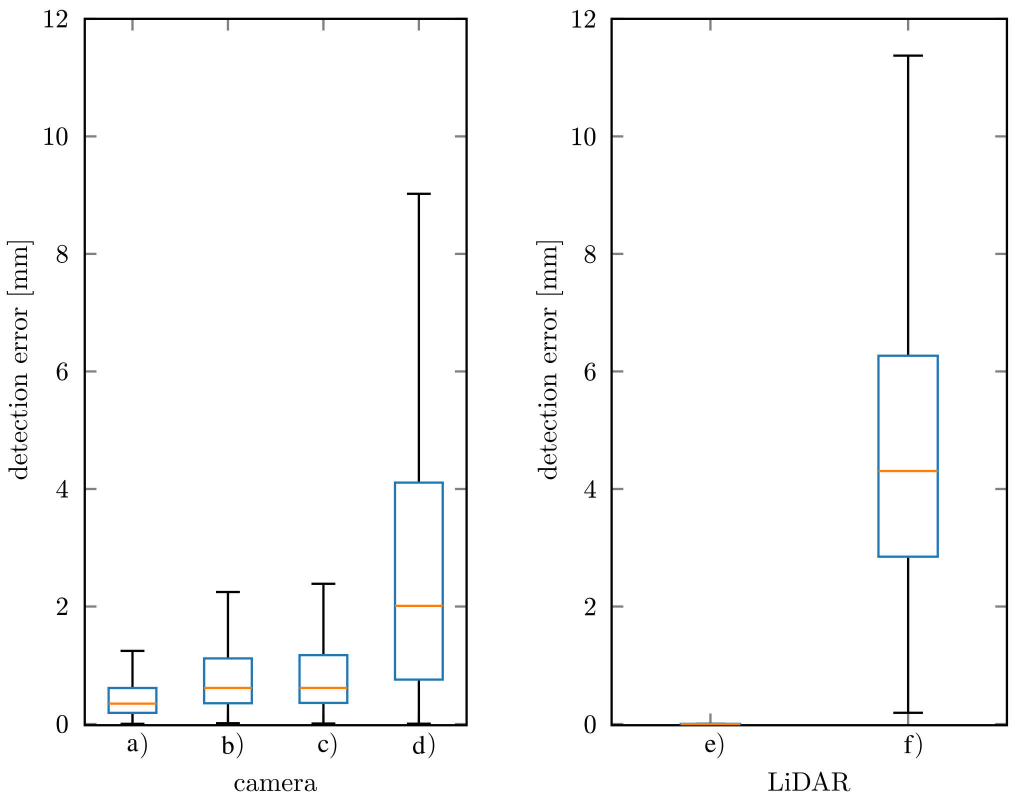


Figure 10.2: Box plots of the detection errors for a camera [a)-d)] and a LiDAR [e)-f)]. In a), the conditions are ideal for camera. The lighting is homogeneous, no blur or pixel noise is added and the background has no texture and a high contrast to the sphere. In b), the sphere is rendered on a realistic background and lighting is not homogeneous. For c), defocus and pixel noise is added. In d), the focal length of the camera is disturbed by 0.3 %. The result shown in e) is generated for point clouds without range noise. The results for realistic range noise is depicted in f). [6]

of $\sigma_{\text{defocus}} = 0.5\,\text{px}$ is appropriate. For the pixel noise, we use a zero-mean Gaussian distribution with a standard deviation of $\sigma_{\text{I}} = 4$ which is emulating our 8 bit grayscale cameras for good lighting. The effect of defocus and pixel noise on the detection error is insignificant. Finally, we simulate an error in the camera intrinsics. The focal length is the worst observable parameter when calibrating a single camera. For good intrinsic calibration, we can assume an error of 0.3 %. The detection accuracy significantly decreases due to this error in focal length. It is by far the most severe error source for the camera sphere detector.

Next, we evaluate the sphere detector for LiDARs. For a LiDAR without range noise, the detections are perfect (see Figure 10.2). The sphere can be perfectly reconstructed independent of the resolution of the LiDAR. This is the great advantage of sampling a sphere compared to estimating edges of a calibration target based on depth jumps. We simulate zero-mean Gaussian noise on the range component with a standard deviation that emulates the measurement noise of modern LiDARs. The detection error varies a lot. This can be explained by the low number of measurements that hit the sphere. We simulate a 3D LiDAR with only 16 scan lines and on average 4 of them hit the sphere. For a LiDAR with higher resolution, the detection error variance is smaller.

This experiment shows how detection errors of a common camera and a common LiDAR compare. A camera that is intrinsically well calibrated can detect the sphere very accurately. A common range sensor usually cannot achieve this level of accuracy. The results show that the detections for cameras are not sensitive to defocus and pixel noise. Further, the texture in the background of the scene has no major influence on the detection accuracy as long as edges still can be detected. Lighting has an influence but is not significant for common scenarios. In our experiments, the most severe error source for camera detections is an inaccurate estimate for the focal length.

10.1.2 Detection Distributions

In the probabilistic calibration framework, we use the assumption that the target center observations are zero-mean Gaussian in the range and angle components. Even if the measurements are zero-mean Gaussian, the observations can be distributed differently due to the detection algorithms. Therefore, we need to check the distributions of the output of our target detectors.

We perform experiments on real data. The biggest challenge when working with real data is to generate ground truth results for comparison. We assume that the distributions of the observations are independent of the sensor resolution up to a scaling factor as long as a minimal amount of measurements lie on the sphere. E.g. for a 4 MP camera and a 1 MP camera, the distributions of the observations are the same up to a scaling factor for the variance. The underlying error sources are the same. This allows us to record data in full resolution to generate ground truth detections which can be used

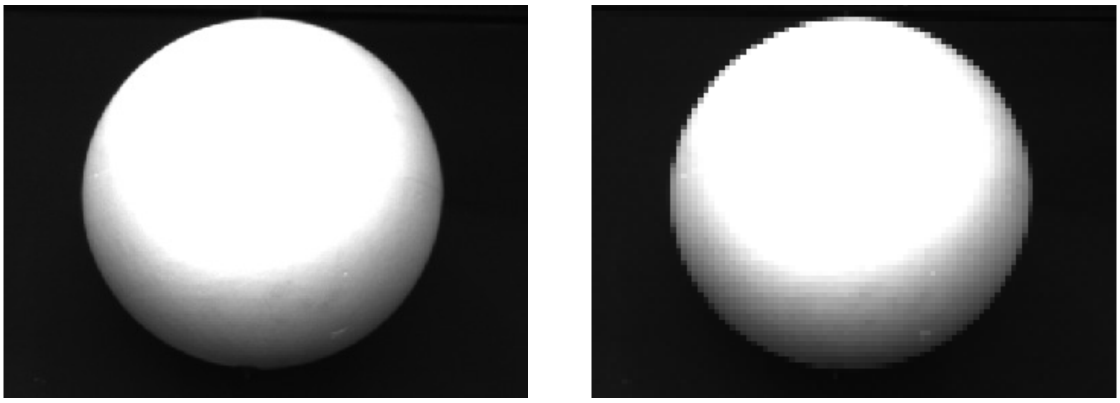


Figure 10.3: Real camera image of the spherical calibration target in full resolution (left) and downsampled to 20 % in each dimension (right). No internal lighting of the sphere is used.

to evaluate detections which are generated on significantly downsampled data. We downsample images to 20 % in each dimension so that the pixel number decreases to 4 % (see Figure 10.3). Point cloud data from a LiDAR with 16 scan lines is randomly downsampled to also 4 % of the measurement points. The calibration target is detected on the full resolution and the downsampled data. The differences $\Delta r, \Delta\theta$ and $\Delta\varphi$ between pairs of high resolution and low resolution detections are calculated for the range r, elevation angle θ and azimuthal angle φ. A dataset of 800 samples is used to generate the histograms in Figure 10.4 and 10.5.

First, let's discuss the results for our LiDAR presented in Figure 10.4. In all components, the histogram can be roughly approximated by a Gaussian distribution. The best Gaussian fits are plotted in orange. They are well centered around zero. The standard deviation of the detections is significantly smaller in the range component than the standard deviation of the range measurement noise of 15 mm. Even for the low number of measurements in the downsampled case, the effect of averaging out the range noise is noticeable.
Next, let's discuss the results for the camera detections in Figure 10.5. We notice that the Gaussian fit in the range error does not model the error distribution well. This makes sense because the range error depends on the distance of the sphere to the camera. Therefore, it makes sense to drop the range information and only use the ray direction for the probabilistic calibration framework which assumes the observations to be Gaussian in all used components. Note that the error due to wrong focal length estimation cannot be analyzed with this

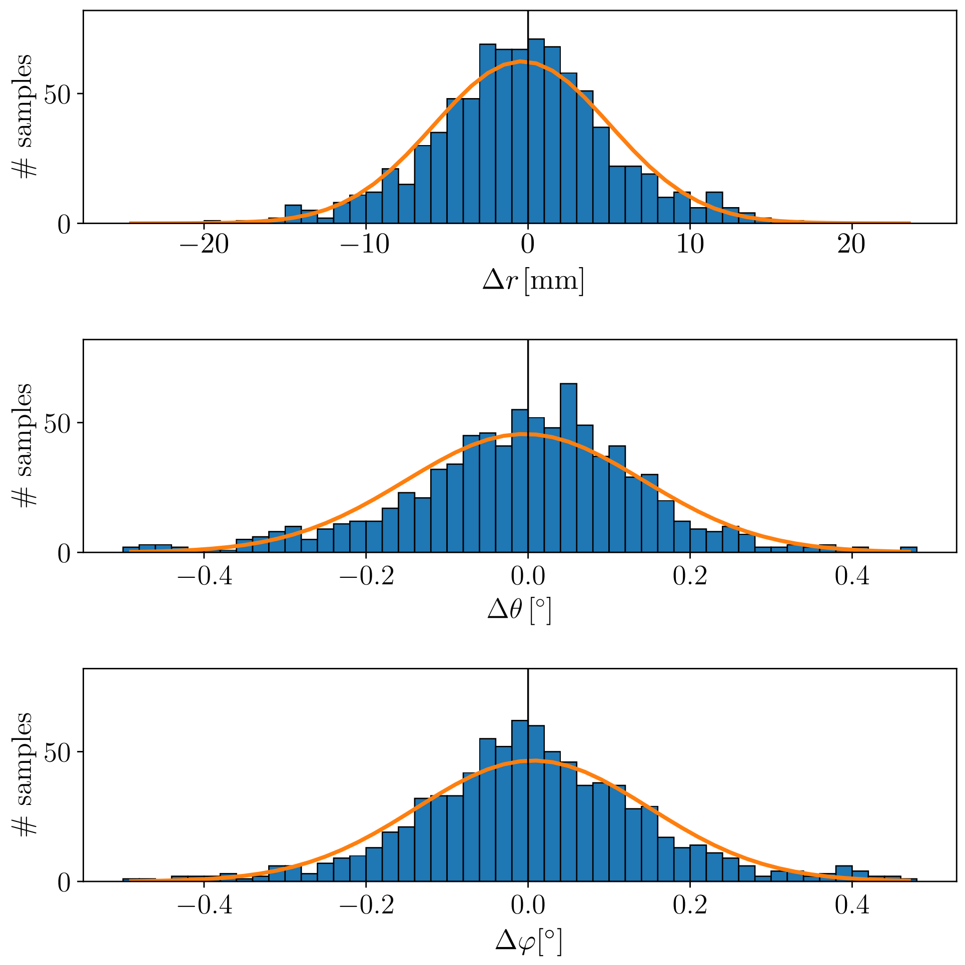


Figure 10.4: Histograms for detection errors in range and angle components for a LiDAR. Additionally, best Gaussian fits are plotted in orange [1].

experiment. The detections on full resolution and low resolution data suffer equally from this problem. The angular components roughly fit the Gaussian assumption. In the elevation angle θ, we notice a small mean shift, whereas in the azimuthal angle φ the mean is centered well. The mean shift in θ is in positive angle direction. The reason for this shift is the dark shadow at the bottom side of the sphere and its bright top side. The dataset is recorded in an indoor scenario where lights are only attached at the ceiling of the room and no light comes in from the windows. Further, no internal lighting of the sphere is used. The images depicted in Figure 10.3 show the lighting conditions in the dataset. The low contrast edge at the bottom in combination with the low resolution results in edge detections which are too close to the sphere center.

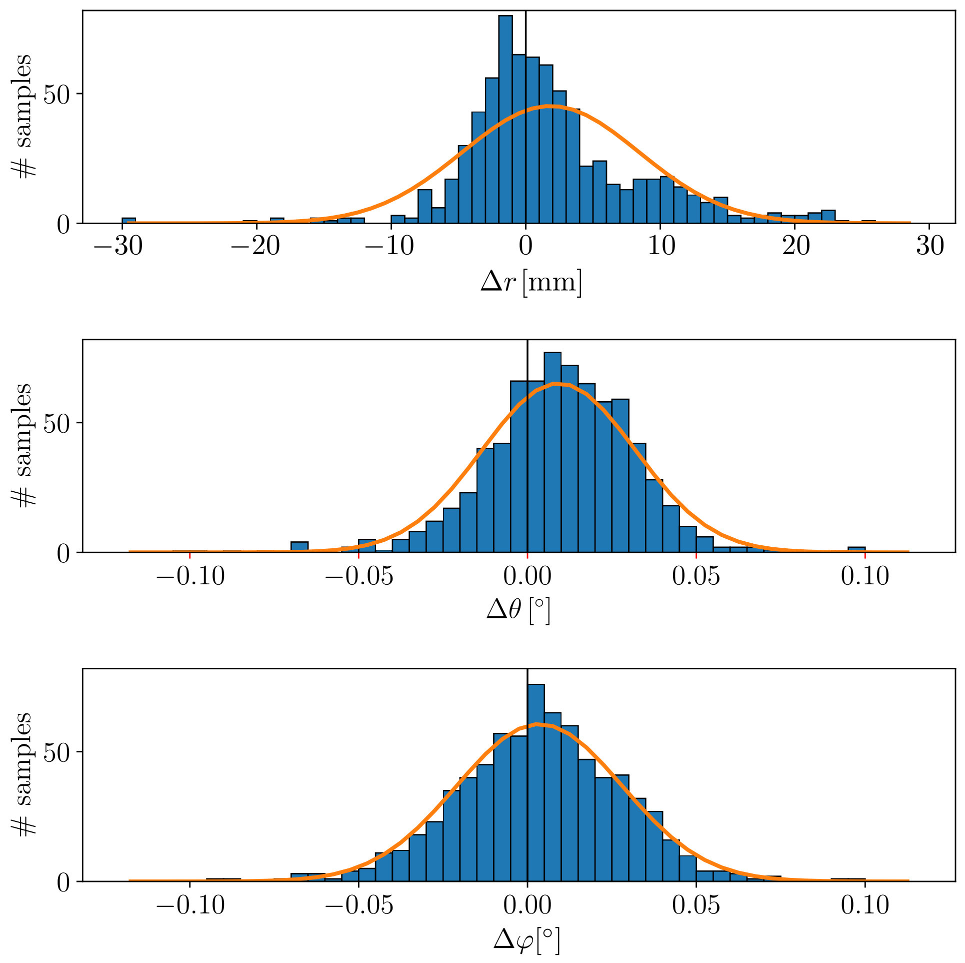


Figure 10.5: Histograms for detection errors in range and angle components for a camera. Additionally, best Gaussian fits are plotted in orange [1].

In azimuthal direction the lighting is symmetric and therefore the mean is centered well.

This experiment shows that the detections are roughly distributed as zero-mean Gaussian in angular components for both camera and LiDAR and, additionally, in range component for LiDAR. The distributions are not perfectly Gaussian and can have minor mean shifts but we will see in section 10.3 that incorporating these uncertainties despite their inaccuracies still improves calibration quality.

10.2 Euclidean Calibration Framework

In this section, the Euclidean calibration framework is evaluated. We start with a sensitivity analysis in simulation to answer the question which error source affects the calibration results the most. Then, real data is used to analyze repeatability and accuracy of the calibration results.

10.2.1 Influence of Different Error Sources

For this experiment, we use our simulation environment again. We can generate observations in different ways. One possibility is to simulate raw sensor data that is passed to the according detectors which provide the observations for the calibration problem. This is done in subsection 10.1.1 to analyze how the detector propagates errors from raw data to observations. In this chapter, we directly simulate the observations. This decouples the calibration optimization problem from the detectors. It allows to analyze the sensitivity of the optimization problem to different errors of the observations. We generate observations directly to analyze three error sources:

First, zero-mean Gaussian noise is added to the observation positions in random directions. We simulate error distances with standard deviation $\sigma_1 = 5\,\mathrm{mm}$ and $\sigma_2 = 10\,\mathrm{mm}$. These values represent upper bounds for realistic detections of a camera and a LiDAR. The values are estimated based on the results of the detector experiments in Figure 10.2.

Second, we analyze the effect of interpolating between observations. Not all sensors provide observations at the same time. Especially rotating sensors cannot provide observations with controlled timestamps. Therefore, we need to interpolate observations to get time synchronized observation pairs for which costs can be defined. To provide realistic results, we simulate a target that moves on a spline with velocity $v = 0.4\,\mathrm{m/s}$. The observation rate is $10\,\mathrm{Hz}$. All simulated observations have different timestamps so that interpolation is needed to generate time synchronized observation pairs.

Third, we analyze the effect of erroneous timestamps. In a professional setup, a time management system is used to synchronize the clocks in the sensors and provides controlled trigger signals to the sensors. In this case, the timestamps have no relevant errors. For cheap or provisional sensor setups, there is no such time management system. Additionally, there are sensors which do not provide the necessary control to manage timestamps. In these cases, sensor data is stamped when it is received by the main processing computer. Then, the error of the timestamps are the sum of the processing time in the sensor and the time to send it to the main computer. To simulate this error, we disturb the perfect timestamps by Gaussian noise. From real data we derive average delay times for cameras of up to 30 ms which varies several milliseconds. Therefore, we use a mean of 30 ms and a standard deviation of 10 ms for the Gaussian distribution on the timestamps of the simulated camera. The data of our LiDAR is sent in small packages so that the processing time and time to send it to the main computer is significantly smaller. In this simulation, we assume the timestamps of the LiDAR observations to be exact.

The results of all three experiments are shown in Figure 10.6. For each experiment, multiple calibration runs for different datasets and different initial poses are performed. The pose errors to the known ground truth poses are split up into translation and rotation errors. The results show multiple interesting points:
First, we get a rough range of the expectable calibration errors. In the experiment a), the simulated noise is higher than what we would expect from real data. Therefore, we can expect pose errors for camera and LiDAR calibration of less than 5 mm in translation and less than 0.1 ° in rotation. We will validate these results on real data in subsection 10.2.2.
Second, we notice from the experiment b) that the errors due to interpolating observations is negligible. Of course, this result depend heavily on the data frequency and the movement of the target. With a data frequency of 10 Hz, the target can be moved with a speed that is convenient for the user while errors are very small. For a data frequency of 1 Hz, the errors can quickly get high if the target is moved on a jerky trajectory.
From the experiment c), we learn that accurate timestamps are essential. A mean time error of 30 ms causes major calibration errors. Again, this error depends on the movement of the target. If multiple static positions of the target are recorded, the result would not suffer from erroneous timestamps. But this would mean significantly more effort to record an appropriately large dataset. Therefore, for a convenient and high quality calibration, a time management system should be used.

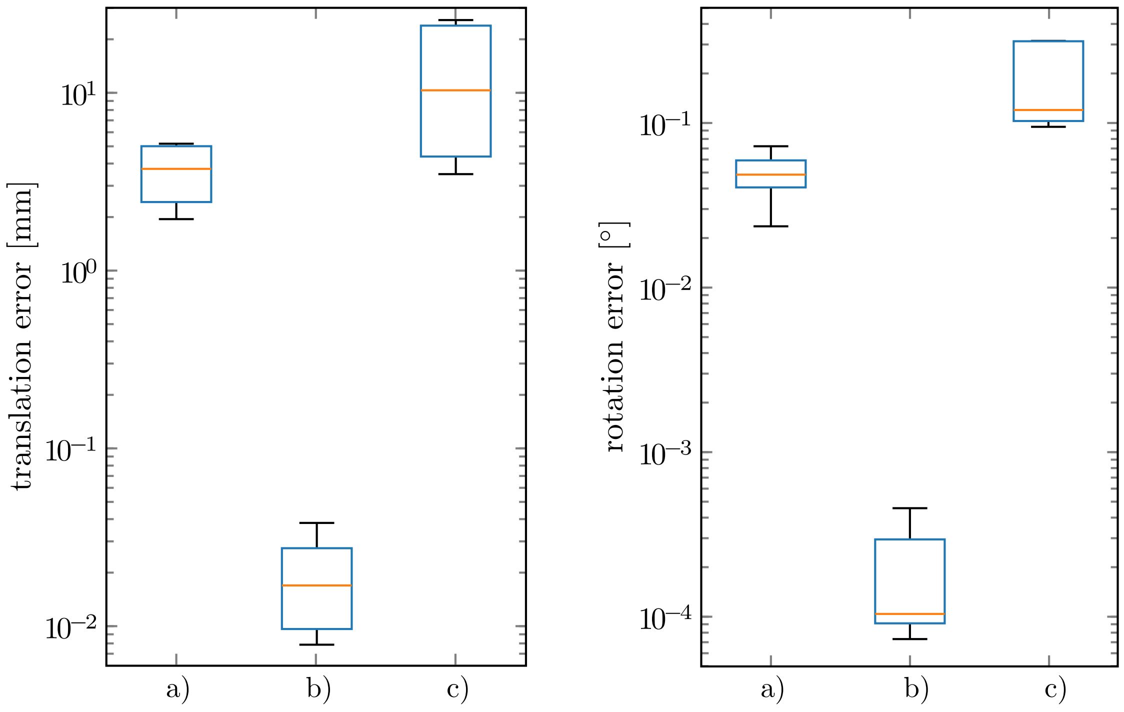


Figure 10.6: Translation and rotation errors of calibration results in three simulated experiments. In (a), the positions of perfect observations are disturbed by zero-mean Gaussian noise. In (b), the error due to interpolating observations at different timestamps is plotted. The error for erroneous timestamps is shown in (c).

10.2.2 Real Data Performance

After the analysis in simulation, we want to check the calibration quality on real data. We fix the pose of a camera and mount a LiDAR on a translation and rotating table to generate a known pose difference to a reference setup (see Figure 10.7). The accuracy of the controlled movement has a tolerance of 0.01 mm in translation and 0.005 ° in rotation and is therefore significantly better than the expected calibration accuracy. For this experiment, we record multiple datasets for three different poses of the LiDAR. The first pose is used as a reference. The second is generated by applying a translation of 120.00 mm to the reference pose. The third is created by rotating the reference pose of the LiDAR by 20.0 °. For each pose, the calibration is performed on multiple datasets and for different initial poses. The results are visualized in Figure 10.8.

First, we analyze the calibration of the reference setup (first row). We notice a median translation difference of 1.4 mm and a median rotation difference

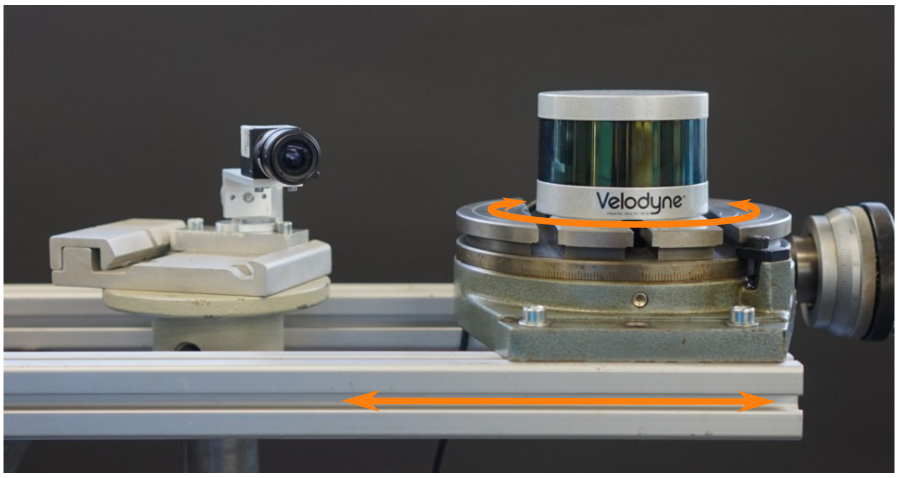


Figure 10.7: Sensor setup consisting of a camera and a LiDAR. The LiDAR can be translated on a linear rail and rotated around its center axis (see orrange arrows) [6].

of 0.07 °. This gives a rough estimate of the repeatability for this setup. In the bottom left, the translation difference is plotted for the case that the LiDAR is shifted by 120 mm. The median translation difference is 122.7 mm which is 2.7 mm off the ideal difference. The results of the calibration runs are not centered around 120 mm which means that systematic errors are present. We notice the same for the results when rotating the LiDAR by 20 ° (bottom right). The median is 19.91 ° which is 0.09 ° off the actual difference. There can be many possible reasons for the systematic errors. One major and always present systematic error is due to imperfect intrinsic sensor models. Sensor models are usually a simplification and cannot model all effects. So, systematic errors are expected on real data, the only question is how large they are. With a median error of 2.7 mm in translation and 0.09 ° in rotation, the errors are sufficiently small for many common applications such as localization and object detection. E.g. in autonomous driving, the detection error of a car in a distance of 100 m is only 16 cm which is sufficient for prediction. Actually, the resulting accuracy and repeatability of our method are significantly better than reported in comparable experiments in literature (e.g. [RFFB08, GMCS12]).

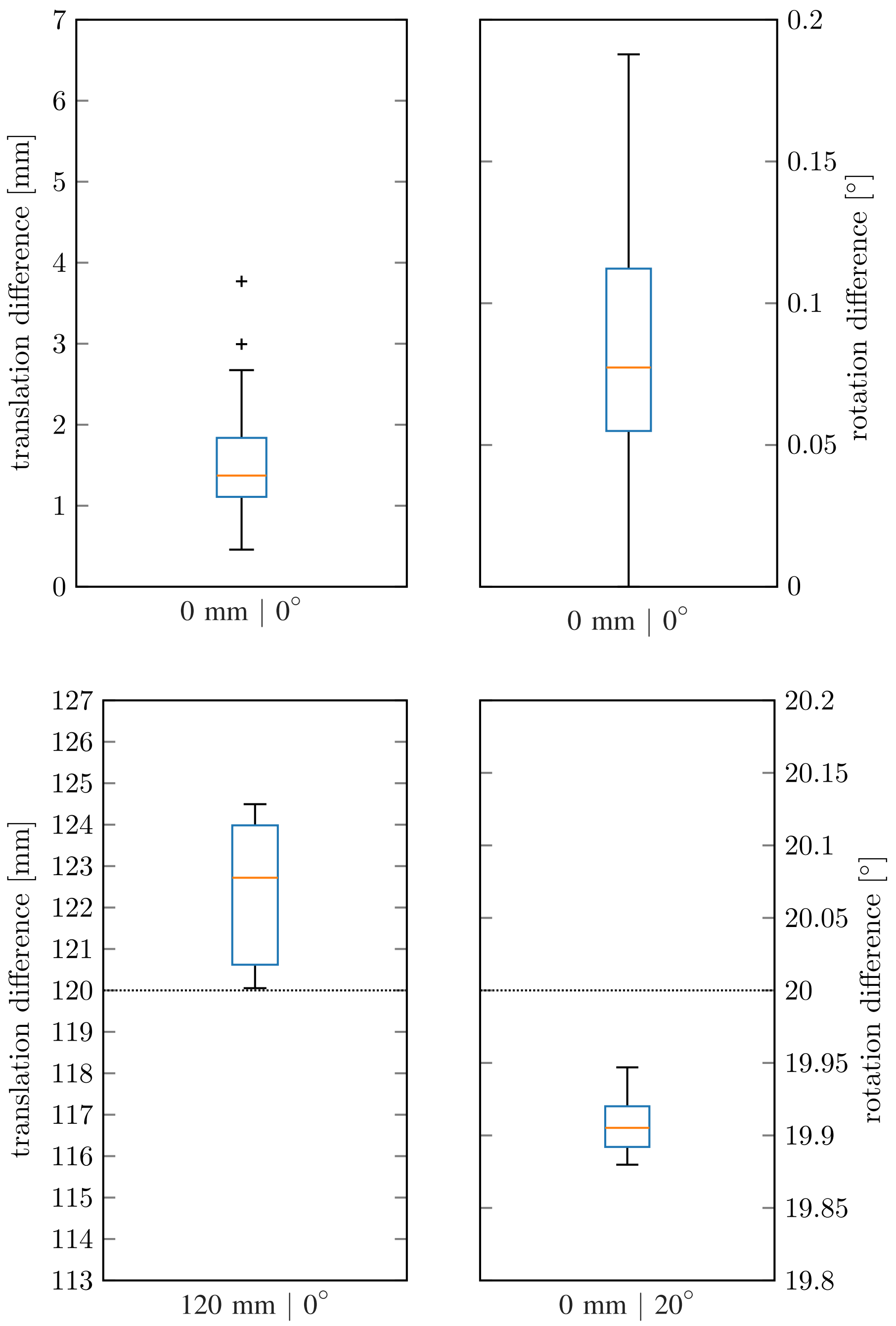


Figure 10.8: A sensor setup consisting of a fixed camera and a movable LiDAR is calibrated multiple times based on different datasets and different initial poses. In the top row, the reference setup is evaluated. In the bottom row left, the LiDAR is moved 120 mm from its reference pose. On the right side of the bottom row, the LiDAR is rotated by 20 ° to its reference pose. The diagrams show the pose differences of each setup to the reference setup [6].

10.3 Probabilistic Calibration Framework

In this section, we evaluate the probabilistic calibration framework and want to answer multiple questions. First, is there a noticeable benefit of using the information of the observation noise? If so, how significant is the benefit and what does it depend on? Further, for practical use, we need to know how sensitive the results are to the observation noise estimates.

10.3.1 Evaluation in Simulation

We start our evaluation in simulation. We generate 1000 independent datasets with given target positions in the range of $2 - 10$ m from the simulated sensors. If the target is in the field of view of a sensor, a noisy observation is created based on the observation noise characteristic of the sensor. On each dataset, several different calibration runs with different settings are performed. By averaging over all 1000 independent datasets the average calibration quality can be assessed. To quantify the quality difference of a target calibration method and a reference calibration method, we introduce the following measure:

$$\begin{aligned} Q(\Delta_{\text{target}}, \Delta_{\text{ref}}) &= \begin{bmatrix} Q_{\text{trans}}(\Delta_{\text{target}}, \Delta_{\text{ref}}) \\ Q_{\text{rot}}(\Delta_{\text{target}}, \Delta_{\text{ref}}) \end{bmatrix} \\ &= \begin{bmatrix} \Delta_{\text{target,trans}} / \Delta_{\text{ref,trans}} \\ \Delta_{\text{target,rot}} / \Delta_{\text{ref,rot}} \end{bmatrix}, \end{aligned} \tag{10.1}$$

where Δ_{target} and Δ_{ref} denote the average differences of the ground truth pose to the pose of a target calibration and to the pose of a reference calibration, respectively. The comparison measurement has two elements, one for the translation and the other for the rotation difference. If both elements are smaller than one, the target method outperforms the reference method. If both elements are larger than one, the reference method outperforms the target method. In the case that one element is larger than one and the other element is smaller than one, no clear assessment can be made because the importance of the accuracy in translation compared to the accuracy in rotation depends on the application.

For the first experiment, we compare the Euclidean to the probabilistic calibra-

	$\text{cam}_{\text{low,ray}}$	$\text{cam}_{\text{high,ray}}$	$\text{cam}_{\text{high,point}}$	$\text{lidar}_{\text{low,point}}$	$\text{lidar}_{\text{high,point}}$
sensor type	camera, low resolution	camera, high resolution	camera, high resolution	LiDAR, low accuracy	LiDAR, high accuracy
observation type	ray	ray	point	point	point
observation noise $(\sigma_r, \sigma_\theta, \sigma_\varphi)$	$(-, 0.05\,°, 0.05\,°)$	$(-, 0.01\,°, 0.01\,°)$	$(0.01\,\text{m}, 0.01\,°, 0.01\,°)$	$(0.1\,\text{m}, 0.1\,°, 0.1\,°)$	$(0.01\,\text{m}, 0.1\,°, 0.1\,°)$

Table 10.1: Definition of simulated sensors.

	$\text{cam}_{\text{low,ray}}$	$\text{cam}_{\text{high,ray}}$	$\text{cam}_{\text{high,point}}$	$\text{lidar}_{\text{low,point}}$	$\text{lidar}_{\text{high,point}}$
$\text{cam}_{\text{high,point}}$	$\begin{bmatrix}0.75\\0.80\end{bmatrix}$	$\begin{bmatrix}0.76\\0.82\end{bmatrix}$	$\begin{bmatrix}0.42\\0.54\end{bmatrix}$	/	/
$\text{lidar}_{\text{low,point}}$	$\begin{bmatrix}0.78\\0.85\end{bmatrix}$	$\begin{bmatrix}0.80\\0.85\end{bmatrix}$	$\begin{bmatrix}0.42\\0.54\end{bmatrix}$	$\begin{bmatrix}0.53\\0.71\end{bmatrix}$	/
$\text{lidar}_{\text{high,point}}$	$\begin{bmatrix}0.75\\0.81\end{bmatrix}$	$\begin{bmatrix}0.77\\0.81\end{bmatrix}$	$\begin{bmatrix}0.77\\0.85\end{bmatrix}$	$\begin{bmatrix}0.60\\0.69\end{bmatrix}$	$\begin{bmatrix}0.52\\0.68\end{bmatrix}$

Table 10.2: Calibration results for different pairs of simulated sensors. Each data cell holds the quality measure $Q(\Delta_{\text{target}}, \Delta_{\text{ref}})$ where we use the probabilistic approach as our target method and the Euclidean calibration approach as the reference method. Duplicates of sensor configurations are marked with '/' [1].

tion framework for different sensor setups. Table 10.1 defines the observation noise of the simulated sensors. The observation noise significantly differs for the sensors. We calibrate different combinations of sensor pairs with the Euclidean calibration framework which ignores the observation noise characteristics and with the probabilistic calibration framework which incorporates the noise information. We use the Euclidean method as the reference method. The target method is the probabilistic calibration framework with exactly known observation noise standard deviations. The results are given in Table 10.2.

At a first glance, we notice that all elements of $Q(\Delta_{\text{target}}, \Delta_{\text{ref}})$ are in every case lower than one. That means that the probabilistic framework outperforms the Euclidean framework in each case. The improvement depends on the sensor pair. We perform the same experiment with setups of three sensors (see Table 10.3). In the first column, we use three highly accurate LiDARs and achieve

$\text{lidar}_{\text{high,point}}$	$\text{lidar}_{\text{high,point}}$	$\text{cam}_{\text{high,point}}$	$cam_{\text{high,point}}$
$\text{lidar}_{\text{high,point}}$	$\text{lidar}_{\text{high,point}}$	$\text{cam}_{\text{high,ray}}$	$\text{cam}_{\text{high,ray}}$
$\text{lidar}_{\text{high,point}}$	$\text{lidar}_{\text{low,point}}$	$\text{cam}_{\text{high,ray}}$	$\text{cam}_{\text{low,ray}}$
$\begin{bmatrix}0.50\\0.69\end{bmatrix}$	$\begin{bmatrix}0.43\\0.69\end{bmatrix}$	$\begin{bmatrix}0.73\\0.79\end{bmatrix}$	$\begin{bmatrix}0.66\\0.72\end{bmatrix}$

Table 10.3: Calibration results for setups of three simulated sensors. Each data cell holds the quality measure $Q(\Delta_{\text{target}}, \Delta_{\text{ref}})$ where we use the probabilistic approach as our target method and the Euclidean calibration approach as the reference method [1].

already large improvements. Next, we replace one of the LiDARs with another LiDAR which has a range noise that is ten times higher (second column). This makes the setup less homogeneous in terms of observation noise and leads to even more improvement. We find the same results for cameras in the third and fourth column where we replace a high with a low resolution camera.
From this experiment, we conclude that taking the information of the observation noise into account leads to significant improvements, especially if the observation noise of the sensors differs a lot.

The previous results are best case scenarios because the true standard deviations are assumed to be known. In reality, the standard deviations have to be estimated. It is important to know how accurate the estimates have to be for good results. Therefore, we again simulate different sensor setups but this time with erroneous observation noise estimates. To generate the erroneous from the true standard deviations, we scale the underlined components in Table 10.4 with γ. The results are compared to the same method with known standard deviations (see Table 10.4). We discuss the experiments in detail:
The first setup (row 2) consists of two sensors that observe the target as a 3D point. The range noise of the first sensor is very high compared to the second sensor. The estimate for the range noise of the first sensor is changed over multiple experiments. The estimate for the angular noise is fixed to the true noise. For an estimate that is too high ($\gamma > 1$), the results are almost as good as with an ideal estimate of the observation noise. The reason is that the range noise of the first sensor is the major error source and estimating it too high decreases its weight in the calibration problem. For an estimate that is too low ($\gamma < 1$), the results get significantly worse. The noisy range observations are trusted more than they should and their weighting in the calibration problem is increased. That is why the calibration result suffers from the noisy range

	$\gamma = \frac{1}{20}$	$\gamma = \frac{1}{10}$	$\gamma = \frac{1}{5}$	$\gamma = \frac{1}{2}$	$\gamma = 2$	$\gamma = 5$	$\gamma = 10$	$\gamma = 20$
$\sigma_1 = (\underline{0.1\,\text{m}}, 0.05\,°, 0.05\,°)$ $\sigma_2 = (0.01\,\text{m}, 0.05\,°, 0.05\,°)$	$\begin{bmatrix}\mathbf{4.47}\\ \mathbf{2.62}\end{bmatrix}$	$\begin{bmatrix}\mathbf{3.54}\\ \mathbf{2.11}\end{bmatrix}$	$\begin{bmatrix}\mathbf{2.06}\\ \mathbf{1.40}\end{bmatrix}$	$\begin{bmatrix}1.04\\ 1.02\end{bmatrix}$	$\begin{bmatrix}1.01\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.01\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.01\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$
$\sigma_1 = (\underline{0.1\,\text{m}}, 0.01\,°, 0.01\,°)$ $\sigma_2 = (0.01\,\text{m}, 0.01\,°, 0.01\,°)$	$\begin{bmatrix}\mathbf{2.23}\\ \mathbf{1.58}\end{bmatrix}$	$\begin{bmatrix}\mathbf{1.58}\\ \mathbf{1.26}\end{bmatrix}$	$\begin{bmatrix}1.10\\ 1.04\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.01\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$
$\sigma_1 = (0.1\,\text{m}, \underline{0.1\,°}, \underline{0.1\,°})$ $\sigma_2 = (0.1\,\text{m}, 0.01\,°, 0.01\,°)$	$\begin{bmatrix}1.02\\ 1.02\end{bmatrix}$	$\begin{bmatrix}1.02\\ 1.01\end{bmatrix}$	$\begin{bmatrix}1.01\\ 1.01\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.04\\ 1.03\end{bmatrix}$	$\begin{bmatrix}\mathbf{2.04}\\ \mathbf{1.40}\end{bmatrix}$	$\begin{bmatrix}\mathbf{4.07}\\ \mathbf{2.43}\end{bmatrix}$	$\begin{bmatrix}\mathbf{6.10}\\ \mathbf{3.71}\end{bmatrix}$
$\sigma_1 = (-, \underline{0.1\,°}, \underline{0.1\,°})$ $\sigma_2 = (0.1\,\text{m}, 0.01\,°, 0.01\,°)$	$\begin{bmatrix}1.04\\ 1.02\end{bmatrix}$	$\begin{bmatrix}1.03\\ 1.01\end{bmatrix}$	$\begin{bmatrix}1.02\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.01\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.00\\ 1.00\end{bmatrix}$	$\begin{bmatrix}1.01\\ 0.99\end{bmatrix}$	$\begin{bmatrix}1.01\\ 1.01\end{bmatrix}$

Table 10.4: Results for comparing the probabilistic method with known standard deviations (reference method) to the probabilistic method with erroneous estimates for the standard deviations (target method). The erroneous standard deviation vector is deduced by multiplying γ to the underlined component in the true standard deviation vector given in the first column. Experiments with four different sensor setups are performed (row 2-5). The quality measure $Q(\Delta_{\text{target}}, \Delta_{\text{ref}})$ is used to quantify the results. We highlight results with a relative error of more than 10 % in at least one component [1].

observations.

The experiments for the second sensor setup (row 3) are similar but the angular noise is smaller than for the first sensor setup. We again notice that for an estimate that is too high ($\gamma > 1$), the results are as good as with an ideal estimate of the observation noise. For an estimate that is too low ($\gamma < 1$), the calibration quality also gets worse but not as quickly as for the first setup. The reason is that the weighting of the angular components is higher for the second setup which means that the range information is overall less important in the calibration problem.

The sensors of the third setup (row 4) have high range noise. The first sensor has high angular noise for which the noise estimation is varied. The angular noise for the second sensor is significantly smaller. Underestimating the high angular noise of the first sensor leads to significant deterioration of the calibration results. The reason is that the angular information is almost ignored. So, the calibration effectively uses only the noisy range information from the first sensor which seems to be a significant information loss. For an estimate that is too low ($\gamma < 1$), the calibration quality does not decrease a lot. This proves that the angular information of the first sensor constrains the problem more significant than the range information.

The last setup (row 5) consists of a sensor that only observes the direction of the target and another sensor that provides point observations. The first sensor has a larger angular noise than the second sensor. When overestimating the angular noise ($\gamma > 1$), the calibration quality almost does not differ from the ideal case. This is due to the second sensor providing significantly more accurate observations so that the target position is essentially estimated by the second sensor. The pose of the first sensor is essentially estimated by minimizing the average angle difference to the fixed target position which is less sensitive to noise. When underestimating the angular noise ($\gamma < 1$), the angular noise of the first sensor is weighted higher so that it is in the dimension of the angular noise of the second sensor. This disturbs the estimate of the target position which leads to less accurate calibration results. But the difference is not large with the given estimates.

This detailed discussion shows that the influence of the observation noise quality is very individual to the sensor setup. But the results in Table 10.4 show that for estimation errors in the range of $0.5 \leq \gamma \leq 2$ the calibration quality does not deteriorate significantly with respect to having exact observation uncertainties. This shows that our method is practical.

10.3.2 Evaluation on Real Data

We want to reproduce the results of the simulation on real data. First, we conduct an experiment with two cameras for a quantitative evaluation. Second, another experiment with more sensor types is used for a qualitative evaluation.

Quantitative Evaluation

For a quantitative evaluation of the probabilistic calibration framework, we use a sensor setup of two cameras. We first calibrate the setup with full resolution images and the complete dataset from which 2000 observations are generated. The result serves as ground truth. Then, the images are downsampled to 20 % in each dimension. Based on this downsampled data, we calibrate the sensor setup again. We use the Euclidean calibration framework and the probabilistic calibration framework with different estimates for the observation noise. First, we use the standard deviation vector $\hat{\boldsymbol{\sigma}}_{\text{real}} = (0.0065\,\text{m}, 0.025\,°, 0.025\,°)$ which we determine from the experiment in subsection 10.1.2 (see Figure 10.5). Then we disturb the angular noise by $\gamma = 5$ and $\gamma = 0.2$. Finally, we use the estimate derived from simulation $\hat{\boldsymbol{\sigma}}_{\text{sim}} = (0.005\,\text{m}, 0.013\,°, 0.013\,°)$. Each calibration method is performed several times by sampling 100 subsets of the complete data. The averaged calibration results are shown in Figure 10.9.

First, we note that the performance of the Euclidean calibration framework is the worst. The best calibration quality is achieved by the probabilistic framework with the observation noise estimate determined through real experiments. The difference is significant in translation and in rotation. Also the error variance (vertical lines) is smaller for the probabilistic framework which shows that it is more reliable than the Euclidean framework. Even for highly disturbed angular noise estimates the probabilistic method outperforms the Euclidean method. This proves that the observation noise estimates do not have to be precisely known for improvements. The noise estimates derived from simulation seem to be also sufficient to reach good results. This has an important practical implication: It shows that simulation of moderate complexity can be used to estimate the observation noise sufficiently well. Simulators can be designed to be very flexible so that different sensors can be simulated without much effort. This replaces real experiments which are more time consuming and costly.

Qualitative Evaluation

Taking the different observation characteristics into account is especially important when calibrating sensors of different types that vary greatly in their observation noise. Therefore, in the following experiment, we present the calibration results for a sensor setup consisting of a camera, two 3D LiDARs and two radars. The camera provides accurate 3D ray observations, the 3D LiDARs observe the target as 3D points and the radars observe 2D points in one plane. The assumed observation variances for each sensor type are given in Table 10.5. Note that the angular accuracy of the radars depends on the distance of the object. Our radars provide measurements with a resolution of 10 cm in the measurement plane. This limits the angular accuracy especially in close distances. Due to the high differences in observation noise of the three sensor types, ignoring the observation characteristics as in the Euclidean calibration framework leads to unreliable results that are often insufficient for

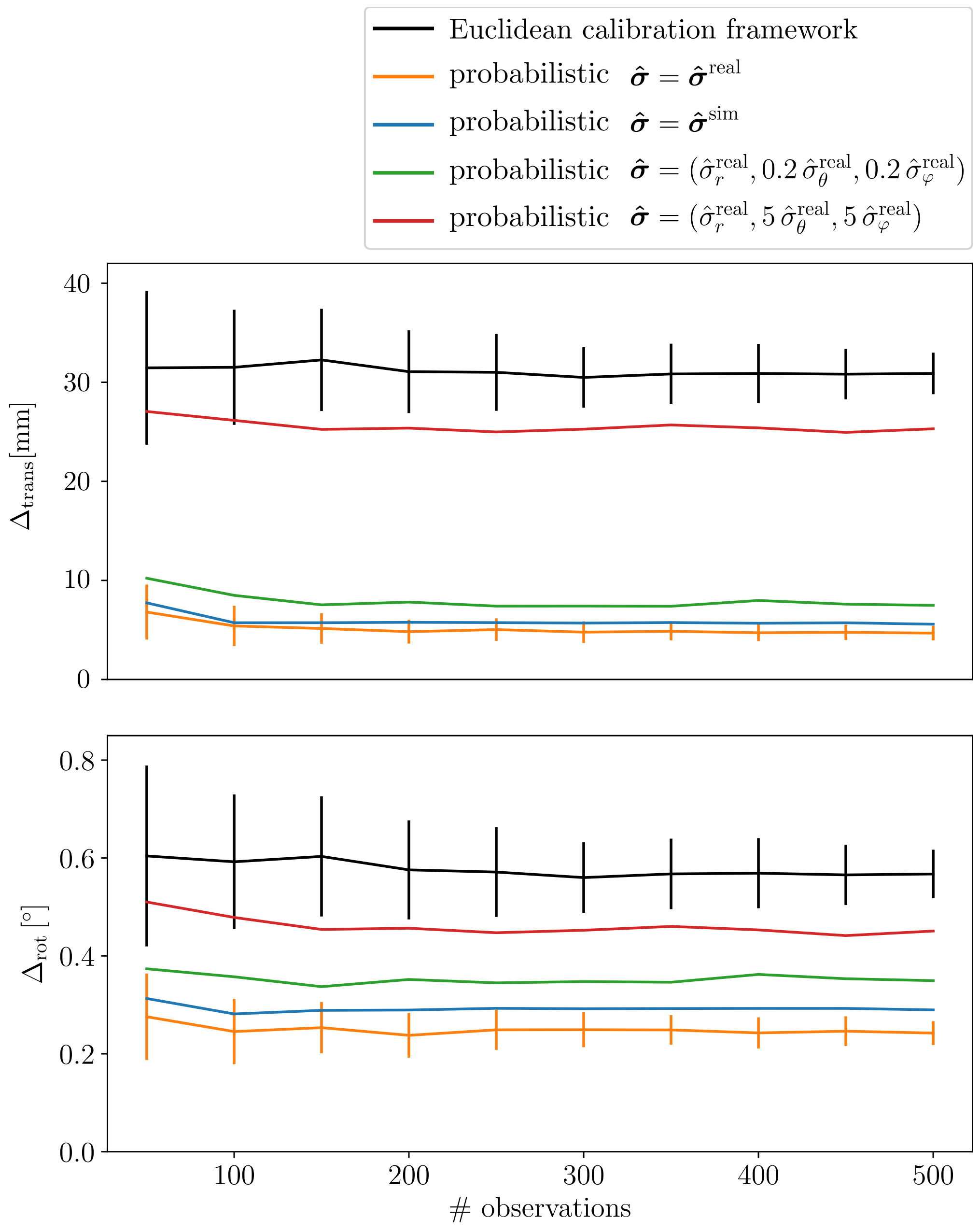


Figure 10.9: Calibration errors for the Euclidean calibration framework and the probabilistic calibration framework with different observation noise estimates. Different noise estimates $\hat{\boldsymbol{\sigma}}$ are used for the proposed method. $\hat{\boldsymbol{\sigma}}^{\text{real}}$ is derived from experiments on real data and $\hat{\boldsymbol{\sigma}}^{\text{sim}}$ is determined based on simulation. Error bars which extend ± 1 standard deviation are shown only for the black and orange curves to preserve clarity [1].

	camera	LiDAR	radar
σ_r	/	0.02 m	0.10 m
σ_θ	0.02 °	0.10 °	0.3 °...1.4 °
σ_φ	0.02 °	0.10 °	/

Table 10.5: Observation noise variances for a setup consisting of a camera, two LiDARs and two radars. '/' marks that the sensor does not observe the according component.

real applications. An application that uses radar, camera and LiDAR data is object detection for autonomous driving. All sensor data is transformed to a reference frame to cluster and combine the information from different sensors in order to create plausible objects. Especially in far distance, calibration errors can lead to failures in associating measurements that fall on the same object.

A simple way for a qualitative calibration assessment is to project point cloud data into the camera image. In Figure 10.10, the noisy data of the two radars is projected to the camera image. Note that the radar measurements are located on a scan plane that is parallel to the ground plane. The camera is mounted roughly one meter above the radar scan plane so that the vertical position of the radar points in the image is a measure for range. Further, the color and the size of the points encode the range as well. To assess the calibration between camera and radars, we look at three areas (marked with numbers) of Figure 10.10 in more detail.
The first cluster of radar measurements is caused by the calibration target. From the overlapping square and the circle, we infer that both radars observe the target. The measurements overlap well and also align in horizontal position with the sphere center in the image. The second cluster of radar measurements is caused by a street lamp. Keeping the resolution of 10 cm in mind, the measurements are projected consistently on the vertical pole of the lamp. The third cluster of radar measurements hits a metal fence. Again, camera and radar data are consistent. From Figure 10.10, we conclude that the calibration between camera and radars was successful. Due to the coarse resolution of the radars, one scene is usually not enough to make a precise assessment of the calibration quality. We noticed good alignment on different scenes. The same process can be used for checking the calibration results for the LiDARs to the camera (see Figure 10.11). As a crosscheck, the point clouds of the radars and the LiDARs are visualized simultaneously in 3D. The overlap of

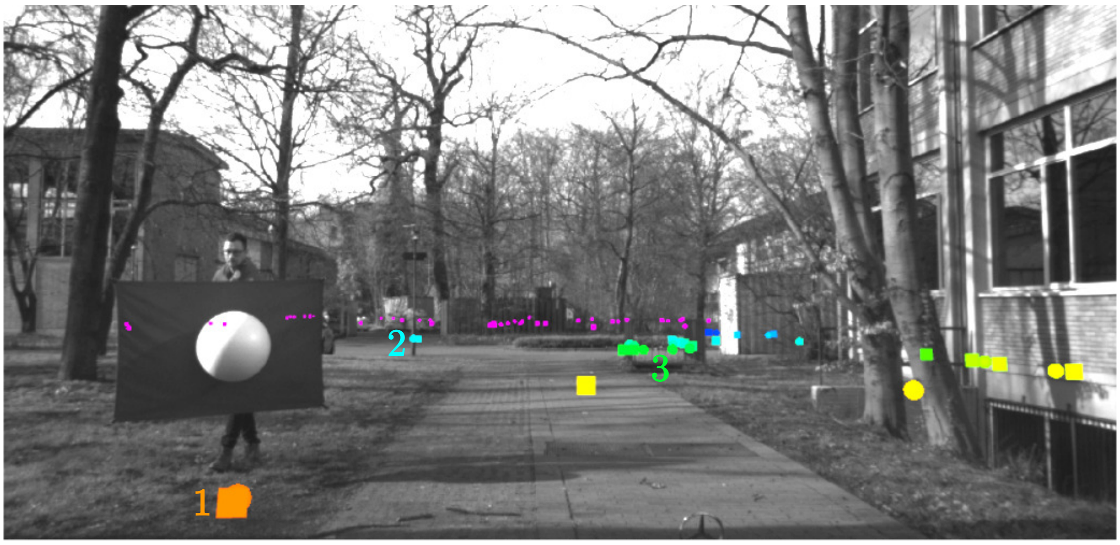


Figure 10.10: Grayscale camera image with projected measurements (colored circles and squares) of two radars after calibration. The radar measurements are located in a plane that is parallel to the ground plane. The camera is mounted roughly one meter above the radars. Since the camera is mounted above the radar, far measurements appear higher in the projection. Additionally, the color and size of the markers encode the range. Squares mark measurements of one radar and circles of the other radar. Note that the point cloud data is provided by the radars with a low resolution of 10 cm. The numbers are used to refer to measurements with the same color.

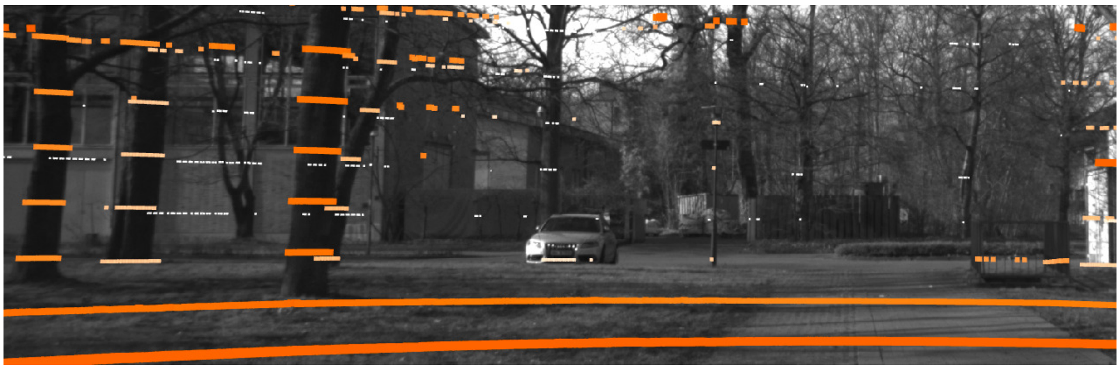
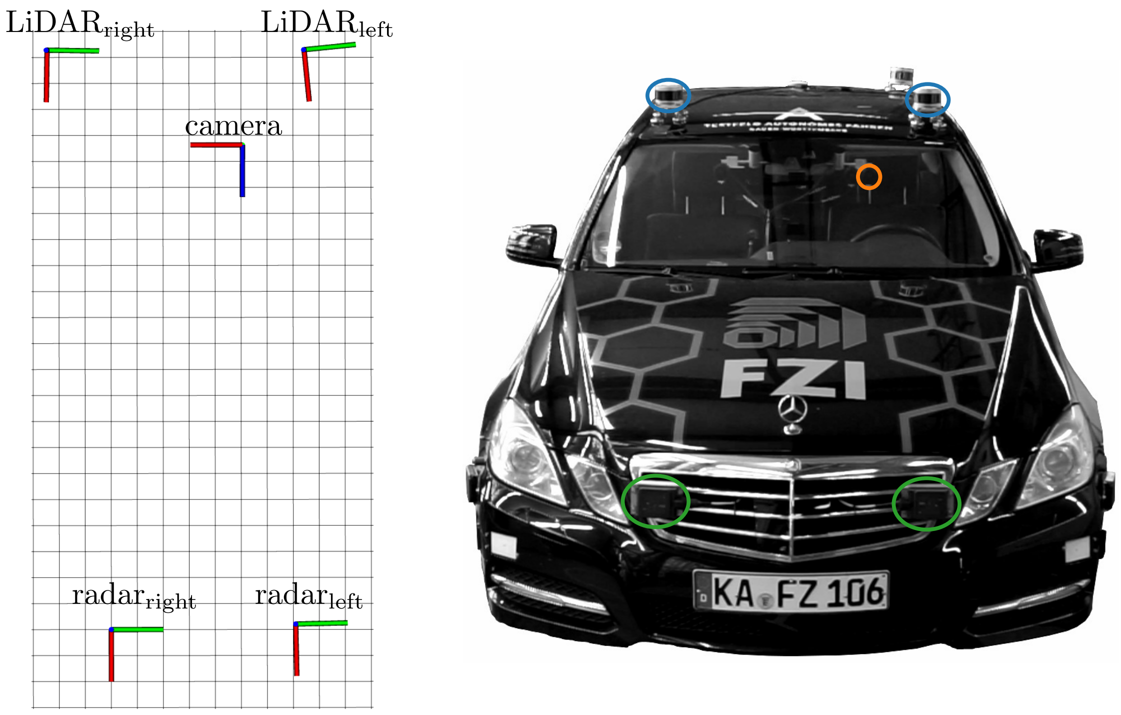


Figure 10.11: Grayscale camera image with projected measurements of a LiDAR after calibration. The color and size of the markers encode the range. An assessment of the camera to LiDAR calibration is possible based on the overlap of projected LiDAR measurements and the image at characteristic objects such as trees and poles.



$\text{LiDAR}_{\text{right}}$
$\text{LiDAR}_{\text{left}}$
camera
$\text{radar}_{\text{right}}$
$\text{radar}_{\text{left}}$
KA FZ 106
FZI

Figure 10.12: Left: Orthographic top-view of the sensor coordinate systems resulting from our calibration. The grid has a cell size of 10 cm × 10 cm. The color code for the coordinate axes is as following: red → x-axis, green → y-axis and blue → z-axis. Right: The calibrated sensor setup mounted on our experimental vehicle. The colors show the sensor types: orange → camera, blue → LiDAR and green → radar.

the point clouds is again evidence for the success of the calibration. As a last check, we can use a measuring tape to check the rough distances between the sensors. Figure 10.12 shows the estimated sensor setup in an orthographic view to the measuring plane of the radars (left) and the real setup (right). The estimated coordinate systems of the sensors are positioned consistently with the real setup.

10.4 Joint Calibration Framework

In this section, we evaluate the joint calibration framework. First, we analyze how range measurements from LiDAR influence the estimation of intrinsic camera parameters. This experiment is conducted on simulated data. Second, we calibrate different sensor combinations with our and a reference method on real data.

10.4.1 Analysis of Camera Intrinsics

First, we want to analyze the effect of adding range measurements to the calibration of intrinsic camera parameters. We simulate a camera and a LiDAR. For the camera, we use the intrinsic model introduced in subsection 3.1.1 with four radial and two tangential distortion parameters and a single viewpoint. Random target trajectories are generated. To simulate camera measurements, the sphere is projected onto the image by using the ground truth intrinsic model. Actually, the inverse of the intrinsic model is needed, which cannot be calculated analytically. Therefore, we use optimization to find the 2D edge point for the according 3D scene point and perform a logic check to assure a reasonable solution is found. The edge measurements are disturbed with Gaussian noise orthogonal to the projected contour. For the measurements of the range sensor, we sample 3D points on the surface of the sphere and add Gaussian noise to the range. We simulate realistic measurement noise by setting the standard deviation for edge detection on camera images to $\sigma_{\text{pix}} = 0.2$ and by using three different range standard deviations $\sigma_{\text{range,low}} = 0.015\,\text{m}$, $\sigma_{\text{range,medium}} = 0.03\,\text{m}$ and $\sigma_{\text{range,high}} = 0.05\,\text{m}$ for three LiDARs.
We perform $n = 1000$ calibration runs with different target trajectories for four types of experiments. For the first experiment, a single camera is calibrated. In the other three experiments, one of the three LiDARs is calibrated with the camera. To analyze and compare the results, we use the mean absolute percentage error $M_{\%}$ and the standard deviation of the absolute percentage error $SD_{\%}$:

$$M_{\%} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{x_i - x}{x} \right| \,, \tag{10.2}$$

$$SD_{\%} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left| 100\% \left| \frac{x_i - x}{x} \right| - M_{\%} \right|^2} \,, \tag{10.3}$$

where x_i is the parameter value estimated in the i-th calibration run and x is the true parameter value. The results for the experiments are shown in Table 10.6.

	camera only	camera + low noise LiDAR	camera + medium noise LiDAR	camera + high noise LiDAR
f	1.21 % (0.97 %)	0.26 % (0.20 %)	0.51 % (0.37 %)	0.75 % (0.59 %)
u_0	0.16 % (0.13 %)	0.13 % (0.09 %)	0.12 % (0.10 %)	0.14 % (0.11 %)
v_0	0.19 % (0.14 %)	0.17 % (0.13 %)	0.16 % (0.12 %)	0.17 % (0.13 %)

Table 10.6: Comparison of intrinsic parameter errors for the calibration of a single camera and camera-LiDAR calibration. The focal length f and the principal point (u_0, v_0) are considered. In each cell, the mean absolute percentage error $M_{\%}$ and the standard deviation of the absolute percentage error $SD_{\%}$ (in brackets) for the considered parameter is calculated over 1000 calibration runs with different target trajectories.

At a first glance, we notice that the mean absolute percentage error and the standard deviation of the absolute percentage error are highest for the single camera calibration. The difference between single camera and simultaneous camera and LiDAR calibration is highest in the focal length. The principal point is also improved by the additional information from LiDAR but significantly less compared to the focal length. Further, we notice that the improvement in the principal point is almost the same for the different LiDARs. The range noise does not significantly influence the estimate. In contrast to that, the focal length substantially depends on the range noise level of the LiDAR measurements. The lower the range noise the better the estimate for the focal length. But even for high range noise, the focal length is significantly improved compared to calibrating the camera alone. The estimation of the target range is inaccurate for calibration with only a single camera. The estimate of the focal length is closely linked to the range estimate of the target. For LiDARs, the range of the target is well observable and can be accurately estimated. Therefore, the LiDAR information complements the intrinsic calibration of a single camera well.

10.4.2 Real Data Performance

In this section, we evaluate the joint calibration framework on real data. Our experimental setup consists of two cameras and two LiDARs (see Figure 10.13). One camera is mounted on a translation table which allows moving it with

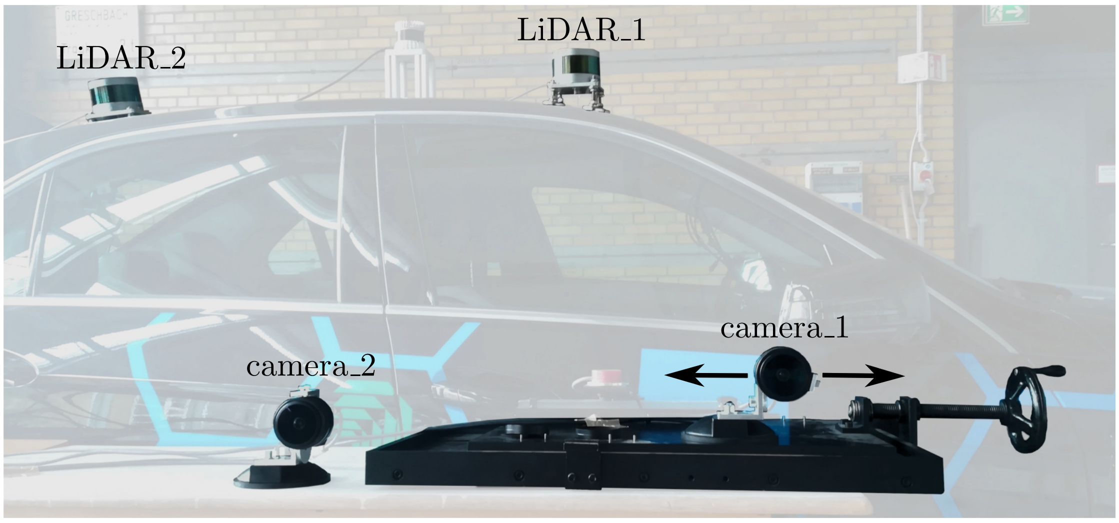


Figure 10.13: Evaluation setup consisting of two LiDARs and two cameras. camera_1 is mounted on a translation table. The other sensors are fixed.

a tolerance better than 0.05 mm. All other sensors are fixed. Both cameras use the same type of fisheye lens with diagonal opening angle of 190 °. The distortion in the images is clearly visible (see Figure 10.14). Our cameras provide 8 MP images. In this experiment, we use the camera setting which is used for our experimental vehicle. The exposure of the sensors is set to automatic mode which leads to suitable brightness of the images over all recorded datasets. The top of the image is cropped so that the resulting image size is 4096×1760 (Figure 10.14 shows the cropped images). This is usually done because the top region only contains irrelevant data for autonomous driving such as sky. Because of the cropping, the principal point is far from the image center, as we will see later.

To assess the quality of our joint calibration framework, we make use of another state of the art calibration method as a reference. The reference method uses checkerboards to calibrate intrinsic and extrinsic parameters of cameras. The method was developed over ten years at our research institute and is considered to be reliable and accurate. Best results are achieved by using the checkerboard frustum shown in Figure 10.14.

We recorded six datasets which we denote with roman numbers I-VI. We generate three different setups by moving camera_1 twice in steps of 50.00 mm. For each setup, we recorded a dataset with the spherical calibration target and

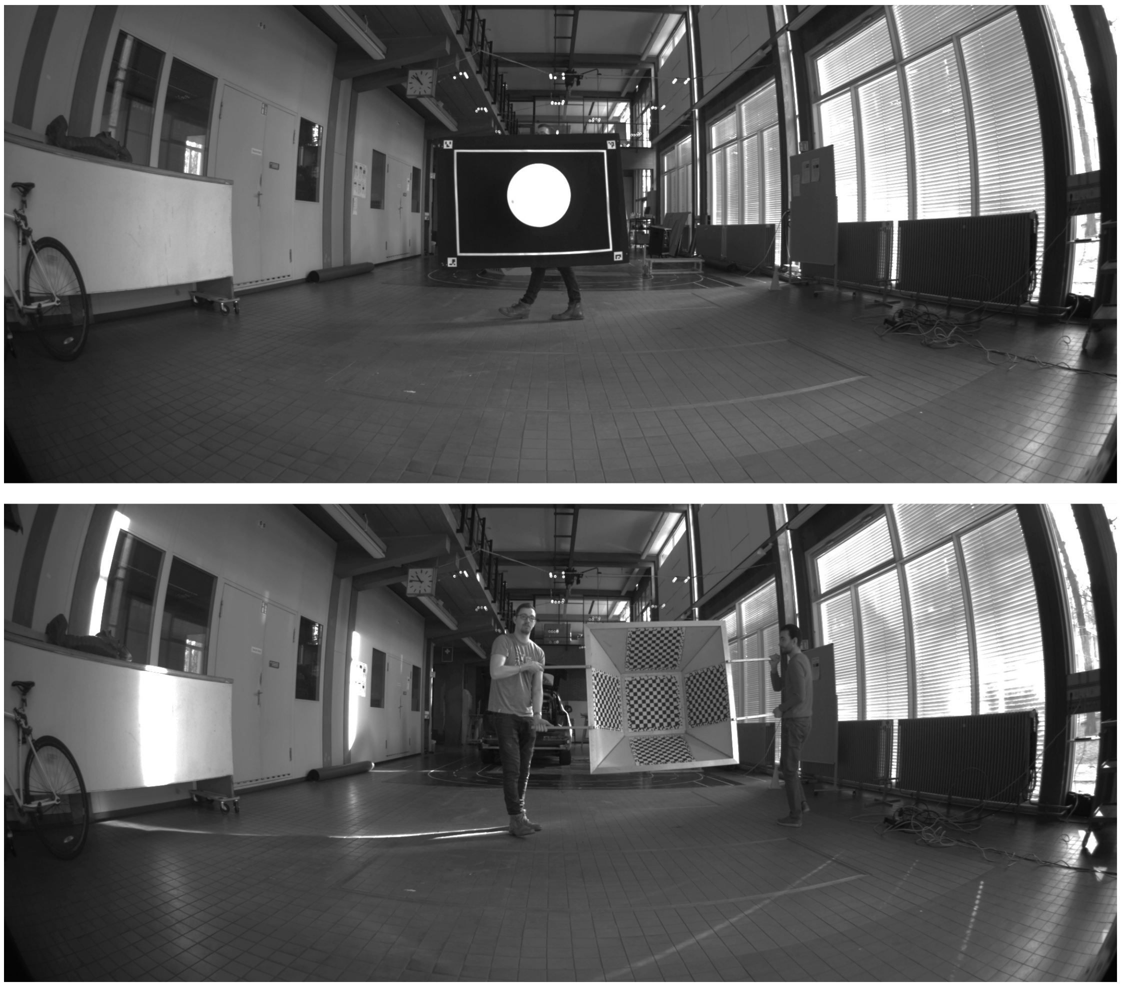


Figure 10.14: Sample images which are used for calibration. In the top image, the spherical calibration target is visible. In the bottom image, a checkerboard target can be seen.

another one with the checkerboard target. Table 10.7 provides an overview of the datasets. Each dataset is about one to three minutes in recording time. For our joint calibration framework, we record at a frequency of 10 Hz for all sensors. This is needed to estimate a smooth spline trajectory based on asynchronous measurements of cameras and LiDARs. For the checkerboard method, all measurements are synchronous because all cameras are triggered at the same time. We use a sensor frequency of 3 Hz so that consecutive images are not too similar to each other. The large checkerboard target provides 605 corners so that reducing the frequency is also a mean to keep calculation time for the calibration reasonable ($<$ 10 min). For the spherical target, we limit the number of edge points to 120 which is similar to the number of corners of a

	I	II	III	IV	V	VI
shift	0.00 mm	0.00 mm	50.00 mm	50.00 mm	100.00 mm	100.00 mm
target	sphere	checker	sphere	checker	sphere	checker

Table 10.7: Overview of the six datasets I-VI used for the evaluation experiments. For each dataset, the shift of the translation table relative to a reference point and the visible calibration target is provided.

single checkerboard on the frustum.
In Figure 10.15, typical images of the coverage of camera measurements are shown for a single dataset with the spherical calibration target and the checkerboard target. First, we see that most parts of the image are well covered by measurements. Note that the sphere is also detected if it is partly outside the view. From the point density, we can see that the targets are detected in different distances to the camera. The lower part of the image is hard to fully cover because of space restrictions. This is a common problem for many setups e.g. for cameras that are mounted behind the windshield of a car. An additional problem is that corners on checkerboards which are tilted too much cannot reliably be detected. We will see in the following evaluation that the coverage is good enough to estimate a camera model of sufficient quality for many applications.

We calibrate different subsets of the sensors. First, we only calibrate a single camera. Then, we add two LiDARs and compare the intrinsics to the calibration results without the LiDARs. The third experiment uses two cameras. Again, the intrinsics are compared to the previous results. Additionally, we check the extrinsics by means of the known position shifts of camera_1. The fourth and final calibration setup consists of two cameras and two LiDARs. Intrinsic and extrinsic parameters are compared to the previous results. An overview of all calibration experiments is given in Table 10.8.

For all calibration experiments, we use the camera model presented in subsection 3.1.1 with four parameters $k_1, \ldots, k_4$ for modeling the radial distortion and two parameters p_1 and p_2 for the tangential distortion. We use an equiangular projection model. Further, we assume a single viewpoint.

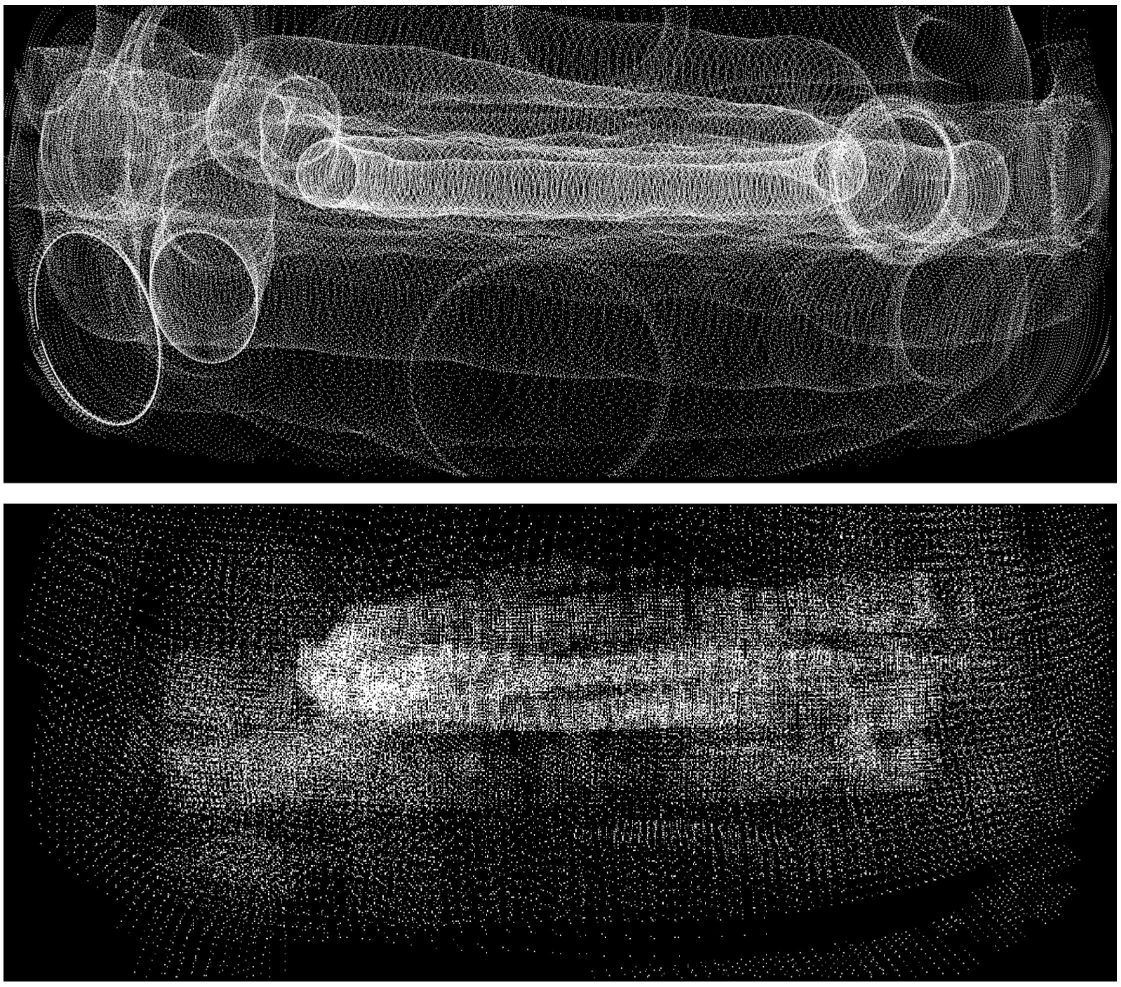


Figure 10.15: Image coverage of camera measurements for a dataset with the spherical calibration target (top) and the checkerboard target (bottom). Each edge or corner measurement is marked with a white dot on the black background.

	single camera	single camera + two LiDARs	two cameras	two cameras + two LiDARs
sphere	×	×	×	×
checkerboard	×	/	×	/

Table 10.8: Overview of evaluation experiments. Crosses mark the conducted experiments. A slash means that the calibration method cannot calibrate all of the used sensors.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
I	2176.61	2058.20	644.02	$15.45 \cdot 10^{-2}$	$14.23 \cdot 10^{-2}$	$-14.18 \cdot 10^{-2}$	$19.03 \cdot 10^{-2}$	$4.43 \cdot 10^{-4}$	$-13.01 \cdot 10^{-4}$
III	1933.16	2065.14	644.37	$9.56 \cdot 10^{-2}$	$11.59 \cdot 10^{-2}$	$-11.40 \cdot 10^{-2}$	$8.83 \cdot 10^{-2}$	$5.46 \cdot 10^{-4}$	$1.94 \cdot 10^{-4}$
V	2021.37	2065.15	641.32	$12.04 \cdot 10^{-2}$	$12.36 \cdot 10^{-2}$	$-13.51 \cdot 10^{-2}$	$12.55 \cdot 10^{-2}$	$-1.06 \cdot 10^{-4}$	$4.20 \cdot 10^{-4}$

Calibration with the joint calibration framework.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
II	1887.36	2061.80	645.04	$9.54 \cdot 10^{-2}$	$8.22 \cdot 10^{-2}$	$-7.42 \cdot 10^{-2}$	$6.21 \cdot 10^{-2}$	$4.67 \cdot 10^{-4}$	$-3.48 \cdot 10^{-4}$
IV	1889.98	2062.19	644.53	$9.28 \cdot 10^{-2}$	$9.34 \cdot 10^{-2}$	$-8.82 \cdot 10^{-2}$	$6.86 \cdot 10^{-2}$	$3.16 \cdot 10^{-4}$	$-3.24 \cdot 10^{-4}$
VI	1890.62	2065.15	644.29	$9.97 \cdot 10^{-2}$	$7.09 \cdot 10^{-2}$	$-5.93 \cdot 10^{-2}$	$5.63 \cdot 10^{-2}$	$3.20 \cdot 10^{-4}$	$0.23 \cdot 10^{-4}$

Calibration with the checkerboard frustum.

Table 10.9: Results for calibrating a single camera with the joint calibration framework (top) and the reference method that uses checkerboards (bottom).

Calibration of a Single Camera

In our first experiment, we only calibrate the intrinsic parameters of a single camera. The results of the joint calibration framework are compared to the results of the reference method that uses checkerboards. We use all six datasets to calibrate the same camera. The datasets I, III and V are used to calibrate with the spherical target and the datasets II, IV, VI are used to calibrate with the checkerboard frustum. The results are shown in Table 10.9. First, let's analyze the results of our calibration method:
The focal length f is estimated unreliably. The results vary in a range of roughly 12 % which is unacceptable. The repeatability on the principal point (u_0, v_0) is significantly better but is still high with a difference of 7 pxl in u_0. The large variance in the radial distortion parameters, especially in the high order parameters k_3 and k_4, shows that the distortion is not estimated reliably. On all three datasets, the average reprojection error is very large (> 1.5 pxl) which shows that the calibration runs were not successful. Before discussing possible reasons, let's look at the results for the checkerboard reference method: The focal length is reliably estimated with a repeatability of better than 0.2 %. The variance in the principal point is less than for the calibration with a sphere. The distortion parameters are in the same range and show significantly less variance than with the previous method. Overall, we conclude that the reference checkerboard method is superior to our method for a single camera, but why?

The first reason for the bad results of our method could be that edge detections at the projected sphere are less accurate than corner detections on the checkerboard pattern. We later show that two cameras can be calibrated accurately by our joint calibration framework. So, the accuracy of the edge detector is not the main reason for the high errors.
A practical reason is that the calibration of a single camera is only an initialization step inside the joint calibration framework. The problem is solved directly without any initialization routine. We always initialize the calibration problem with vanishing distortion parameters, a principal point at the center of the image and a focal length in a range of 10 % from the true value. If the calibration problem is initialized with better distortion parameters and a better focal length, the calibration with our method is significantly more reliable. So, we see potential for improvement.
Another major difference between the spherical calibration target and the checkerboard frustum is that, for the sphere, measurements only lie on the edge of its projection, whereas the checkerboards generate detections well distributed over a large area of the image. We suspect that the distortion parameters are better constrained by the well distributed corner detections of the checkerboard than by measurements on a single edge contour of a sphere. This could be further investigated by creating a custom calibration board with corners only on a single contour (e.g. on a circle).
Calibration of a single camera with checkerboards is well established and shows high accuracy, so, there is no need to replace this standard method by another approach. Therefore, calibrating a single camera is not the main focus of this work. Our method is designed for multimodal calibration, which is why we proceed with the next experiment that uses a single camera in combination with LiDARs.

Calibration of a Single Camera and LiDARs

We calibrate the same camera as in the previous section, but this time, we add two LiDARs. The results for the camera intrinsics are shown in Table 10.10.
We clearly see a significant improvement in repeatability of the focal length and the radial distortion parameters compared to calibrating a single camera with our method. We also reach a higher repeatability than the checkerboard method for a single camera. The estimation accuracy of the principal point is in the same range as for calibrating the single camera with our method.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
I	1893.35	2059.33	645.02	$9.53 \cdot 10^{-2}$	$7.99 \cdot 10^{-2}$	$-6.94 \cdot 10^{-2}$	$6.14 \cdot 10^{-2}$	$5.13 \cdot 10^{-4}$	$-9.47 \cdot 10^{-4}$
III	1893.98	2062.60	642.92	$9.77 \cdot 10^{-2}$	$7.89 \cdot 10^{-2}$	$-7.18 \cdot 10^{-2}$	$6.33 \cdot 10^{-2}$	$3.78 \cdot 10^{-4}$	$-3.43 \cdot 10^{-4}$
V	1894.01	2064.03	640.58	$9.79 \cdot 10^{-2}$	$8.19 \cdot 10^{-2}$	$-7.59 \cdot 10^{-2}$	$6.47 \cdot 10^{-2}$	$-1.91 \cdot 10^{-4}$	$2.03 \cdot 10^{-4}$

Table 10.10: Results for calibrating a single camera and two LiDARs with the joint calibration framework.

These results are consistent with the experiments on simulated data (see subsection 10.4.1).

The LiDAR data provides additional range information which is beneficial for estimating the range of the calibration target. Since the range of the target and the focal length are closely linked, the focal length becomes well observable by adding the LiDAR data. Therefore, the estimate of the focal length is significantly improved. Radial distortion and focal length are also linked. An erroneous focal length estimation can be partly compensated by a wrong radial distortion. This coupling is clearly noticeable when calibrating a single camera with our method (see Table 10.9). In this experiment with a single camera and two LiDARs, the radial distortion can be estimated with high repeatability due to the well observable focal length.

Calibration of Two Cameras

Next, we calibrate two cameras. The resulting intrinsic parameters for our and the reference method are given in Table 10.11.
Our joint calibration framework shows high repeatability in focal length. The result is comparable to the previous experiment with a single camera and two LiDARs. The focal length seems to be significantly better observable than when calibrating a single camera. This makes sense, since the range of the target can be better estimated by two cameras compared to a single camera due to the different viewpoints. Additionally, the principal point shows improved repeatability compared to calibrating a single camera with LiDARs. Actually, the repeatability is in the range of the reference method.

Next, we analyze the estimated extrinsic parameters. We use the translation table to move camera_1 twice by 50.00 mm. As shown in Table 10.7, we

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
I	1896.98	2065.22	639.59	$10.07 \cdot 10^{-2}$	$6.06 \cdot 10^{-2}$	$-3.80 \cdot 10^{-2}$	$4.61 \cdot 10^{-2}$	$-2.16 \cdot 10^{-4}$	$0.42 \cdot 10^{-4}$
III	1896.62	2066.53	640.10	$9.61 \cdot 10^{-2}$	$8.06 \cdot 10^{-2}$	$-6.79 \cdot 10^{-2}$	$6.04 \cdot 10^{-2}$	$-0.47 \cdot 10^{-4}$	$3.73 \cdot 10^{-4}$
V	1897.41	2066.54	638.53	$9.84 \cdot 10^{-2}$	$8.05 \cdot 10^{-2}$	$-7.27 \cdot 10^{-2}$	$6.39 \cdot 10^{-2}$	$-4.35 \cdot 10^{-4}$	$6.12 \cdot 10^{-4}$

Intrinsic parameters for camera_1 resulting from calibration with the joint calibration framework.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
II	1891.47	2062.20	641.22	$9.59 \cdot 10^{-2}$	$8.23 \cdot 10^{-2}$	$-7.37 \cdot 10^{-2}$	$6.24 \cdot 10^{-2}$	$-0.85 \cdot 10^{-4}$	$-2.34 \cdot 10^{-4}$
IV	1892.88	2062.15	640.44	$9.47 \cdot 10^{-2}$	$8.98 \cdot 10^{-2}$	$-8.44 \cdot 10^{-2}$	$6.78 \cdot 10^{-2}$	$-2.21 \cdot 10^{-4}$	$-2.99 \cdot 10^{-4}$
VI	1893.31	2064.35	640.59	$9.98 \cdot 10^{-2}$	$7.21 \cdot 10^{-2}$	$-6.06 \cdot 10^{-2}$	$5.71 \cdot 10^{-2}$	$-1.46 \cdot 10^{-4}$	$-0.62 \cdot 10^{-4}$

Intrinsic parameters for camera_1 resulting from calibration with the reference method.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
I	1894.59	2072.66	698.82	$9.50 \cdot 10^{-2}$	$8.14 \cdot 10^{-2}$	$-6.92 \cdot 10^{-2}$	$6.16 \cdot 10^{-2}$	$-0.56 \cdot 10^{-4}$	$-3.89 \cdot 10^{-4}$
III	1894.37	2074.89	698.96	$8.80 \cdot 10^{-2}$	$10.88 \cdot 10^{-2}$	$-10.66 \cdot 10^{-2}$	$7.77 \cdot 10^{-2}$	$-0.53 \cdot 10^{-4}$	$-0.64 \cdot 10^{-4}$
V	1895.46	2076.43	697.64	$9.76 \cdot 10^{-2}$	$7.91 \cdot 10^{-2}$	$-6.79 \cdot 10^{-2}$	$6.05 \cdot 10^{-2}$	$-4.24 \cdot 10^{-4}$	$3.19 \cdot 10^{-4}$

Intrinsic parameters for camera_2 resulting from calibration with the joint calibration framework.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
II	1889.37	2074.25	700.00	$9.18 \cdot 10^{-2}$	$9.68 \cdot 10^{-2}$	$-9.43 \cdot 10^{-2}$	$7.18 \cdot 10^{-2}$	$-0.63 \cdot 10^{-4}$	$-0.93 \cdot 10^{-4}$
IV	1891.26	2074.64	699.30	$9.43 \cdot 10^{-2}$	$9.10 \cdot 10^{-2}$	$-8.68 \cdot 10^{-2}$	$6.94 \cdot 10^{-2}$	$-2.03 \cdot 10^{-4}$	$-1.49 \cdot 10^{-4}$
VI	1891.66	2076.02	699.87	$10.01 \cdot 10^{-2}$	$6.70 \cdot 10^{-2}$	$-5.09 \cdot 10^{-2}$	$5.18 \cdot 10^{-2}$	$-0.50 \cdot 10^{-4}$	$0.47 \cdot 10^{-4}$

Intrinsic parameters for camera_2 resulting from calibration with the reference method.

Table 10.11: Intrinsic parameters resulting from calibrating two cameras with the joint calibration framework and the checkerboard method. The top two tables show the results for camera_1 and the bottom two tables show the results for camera_2.

recorded datasets with the spherical and checkerboard target for each position of the shifted camera. This allows us to estimate the shifts with both methods. We use the known shifts to assess the quality of the estimated positions of the cameras. Table 10.12 a)-b) shows the estimated shifts and the errors for our and the reference method. We note that both approaches can estimate the shifts with sub-millimeter accuracy. Dataset III shows the highest error with 0.38 mm. On these datasets, the average error of estimating the shift is higher for our method compared to the checkerboard method.

Calibration of Two Cameras and LiDARs

Finally, we calibrate two cameras and two LiDARs with our joint calibration framework. The resulting intrinsic parameters for both cameras are given in

	est. shift [mm]	error [mm]
I	/	/
III	49.62	0.38
V	100.04	0.04

a)

	est. shift [mm]	error [mm]
II	/	/
IV	50.08	0.08
VI	99.95	0.05

b)

	est. shift [mm]	error [mm]
I	/	/
III	49.62	0.38
V	100.02	0.02

c)

Table 10.12: Estimated shifts of camera_1 relative to the reference position in datasets I and II. In a), the two cameras are calibrated with the joint calibration framework. b) shows the results with the reference method. In c), two cameras and two LiDARs are calibrated with a sphere.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
I	1896.45	2065.54	639.46	$9.95 \cdot 10^{-2}$	$6.48 \cdot 10^{-2}$	$-4.51 \cdot 10^{-2}$	$4.98 \cdot 10^{-2}$	$-2.24 \cdot 10^{-4}$	$0.63 \cdot 10^{-4}$
III	1896.35	2066.31	639.87	$9.64 \cdot 10^{-2}$	$7.97 \cdot 10^{-2}$	$-6.70 \cdot 10^{-2}$	$5.99 \cdot 10^{-2}$	$-0.75 \cdot 10^{-4}$	$3.31 \cdot 10^{-4}$
V	1897.12	2066.43	638.40	$9.83 \cdot 10^{-2}$	$8.04 \cdot 10^{-2}$	$-7.26 \cdot 10^{-2}$	$6.38 \cdot 10^{-2}$	$-4.49 \cdot 10^{-4}$	$5.95 \cdot 10^{-4}$

Intrinsic parameters for camera_1.

	f [pxl]	u_0 [pxl]	v_0 [pxl]	k_1	k_2	k_3	k_4	p_1	p_2
I	1894.20	2072.63	698.64	$9.49 \cdot 10^{-2}$	$8.09 \cdot 10^{-2}$	$-6.90 \cdot 10^{-2}$	$6.15 \cdot 10^{-2}$	$-0.74 \cdot 10^{-4}$	$-4.01 \cdot 10^{-4}$
III	1894.07	2074.80	698.72	$8.81 \cdot 10^{-2}$	$10.83 \cdot 10^{-2}$	$-10.60 \cdot 10^{-2}$	$7.75 \cdot 10^{-2}$	$-0.78 \cdot 10^{-4}$	$-0.86 \cdot 10^{-4}$
V	1895.19	2076.42	697.50	$9.78 \cdot 10^{-2}$	$7.84 \cdot 10^{-2}$	$-6.71 \cdot 10^{-2}$	$6.01 \cdot 10^{-2}$	$-4.40 \cdot 10^{-4}$	$3.16 \cdot 10^{-4}$

Intrinsic parameters for camera_2.

Table 10.13: Intrinsic parameters resulting from calibrating two cameras and two LiDARs with the joint calibration framework.

Table 10.13. The results are comparable to the results when calibrating two cameras. This means that the additional LiDAR data does not significantly improve the calibration quality if two cameras are used. The range of the target is already well estimated by the cameras.
At this point, we want to check if the estimated distortion models are suitable. Figure 10.16 shows images that are projections of the raw images on a simple pinhole model by means of the estimated parameters of our (top) and the reference method (bottom). If the estimated camera models are suitable, straight lines in the scenes are projected on the images as straight lines. This is the case for both estimated camera models. Actually, differences in the undistorted images are hardly visible. Our and the reference method can both estimate suitable camera models for this camera setup.

We want to analyze the results of the joint calibration framework in more detail. A common practice is to look at the distributions of the residuals.

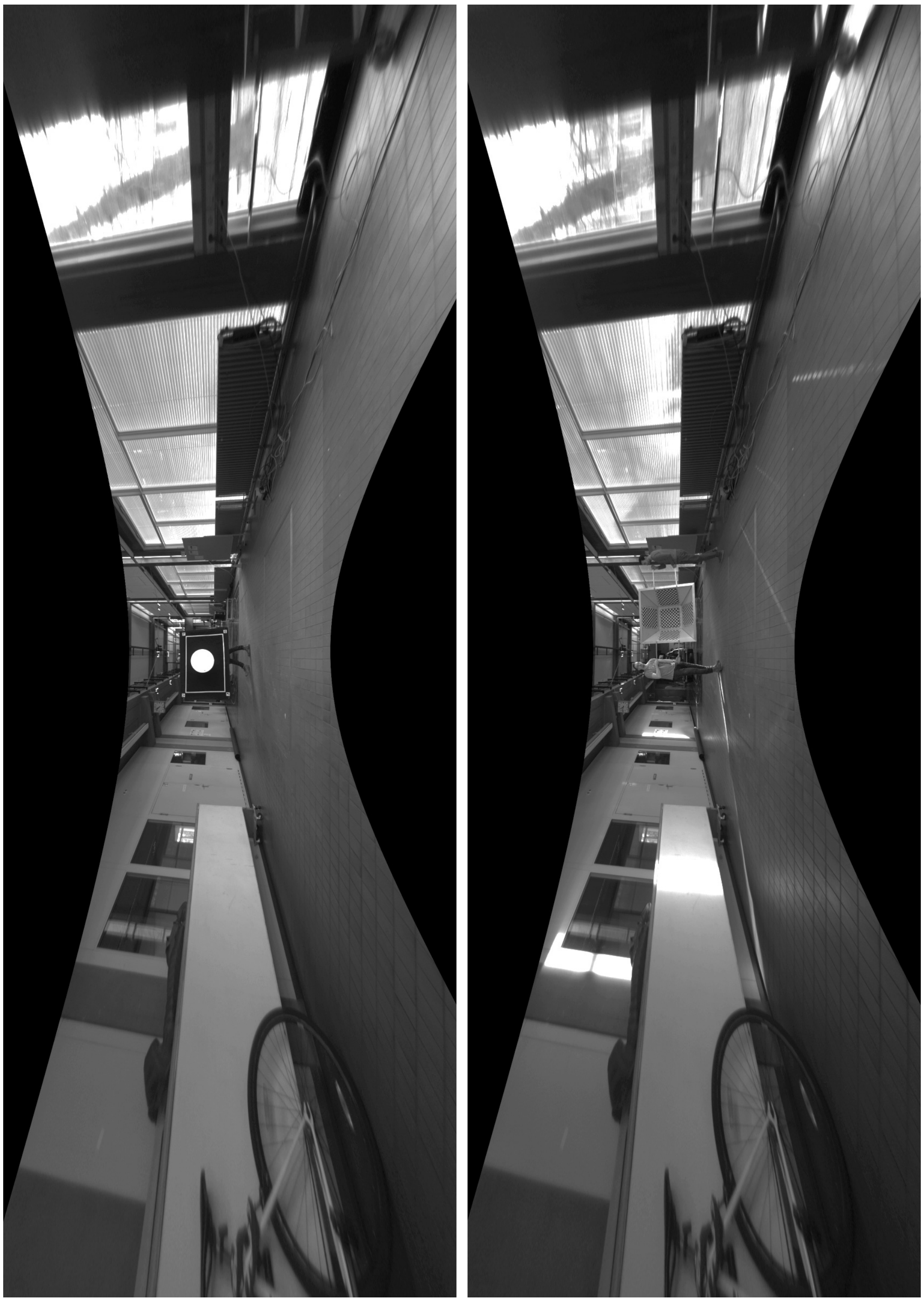


Figure 10.16: The raw images of Figure 10.14 are undistorted and projected using a pinhole model. For the top image, the intrinsics resulting from our joint calibration framework are used. The bottom image is generated based on the intrinsics resulting from the checkerboard method.

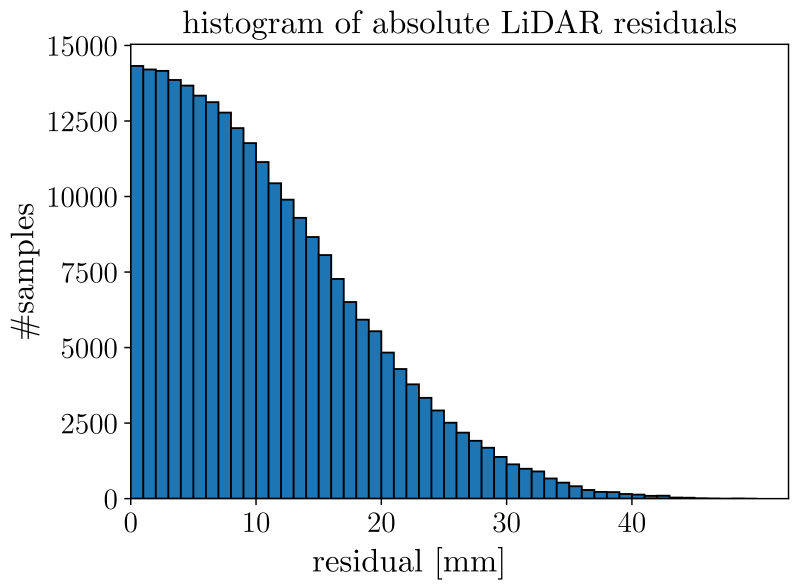


Figure 10.17: Absolute residual distribution of LiDAR_1.

Figure 10.17 shows the distribution of the residuals for LiDAR_1. The maximum sample peak is clearly at zero and rapidly decreases with rising residuals. The absolute residual mean is 10.8 mm which is in the expected range. The range accuracy given by the datasheet is ±30 mm which is consistent with the residuals distribution. The residual distribution for LiDAR_2 is similar.
The distribution of the camera residuals is shown in Figure 10.18. The residuals are split up in their normal and tangential component. The distribution is centered well and symmetric in both components. The normal component has an absolute mean of 0.25 mm, whereas the tangential component has a significantly higher absolute mean of 2.49 mm. The reason for this difference is that the estimated noise of the tangential component, caused by the noise of the tangent angle estimation, is significantly higher than the noise of the normal component. Since the noise distributions are considered in the joint calibration framework, the normal component is weighted significantly higher in the optimization problem than the tangential component. This is why the average mean of the tangential component is higher than the average mean of the normal component.
All residual distributions indicate a successful calibration. Small residuals that are symmetrically distributed are only a necessary but not a sufficient condition for a successful calibration. Therefore, we want to analyze another quality measure that is not directly optimized for. We define a reprojection

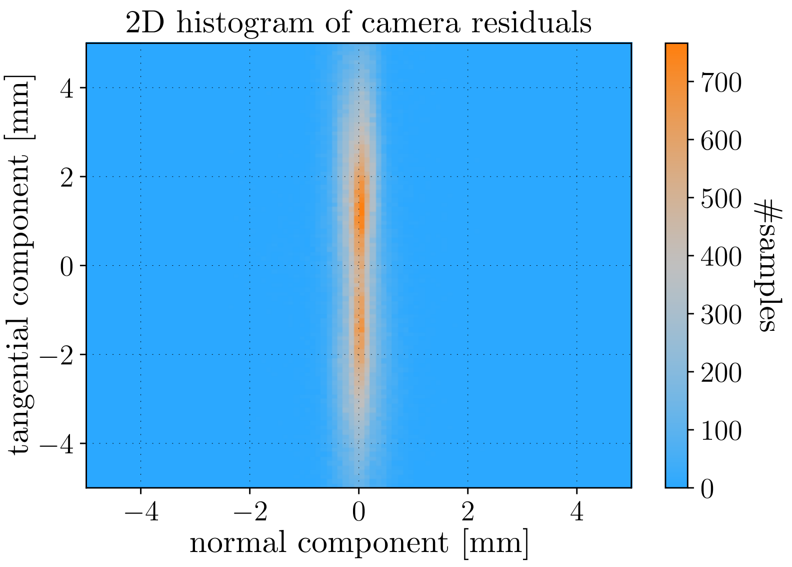


Figure 10.18: Residual distribution of camera_1 split up in its tangential and normal component.

error in the 2D image space for our spherical target as following:
For a viewing ray corresponding to an edge point measurement on the image, we search the closest point on the estimated sphere. The closest point is projected on the image. We define the pixel distance between this projected point and the edge point measurement as our reprojection error for the spherical target. Note that this definition is significantly different from the residual definition in which the tangent direction is used additionally to the edge position. This makes the reprojection error a suitable measure to assess the quality of the calibration.
Figure 10.19 shows 2D and 1D histograms of the reprojection errors resulting from calibrating with the spherical target and the checkerboard. First, we notice that the directional distribution is not perfectly symmetrical with our method. Possible reasons for that might be systematic errors in the edge detections and in the LiDAR data. However, the distribution is centered and the average absolute error is small with only 0.20 pxl. The checkerboard method results in a more symmetric distribution. Its high corner detection accuracy is evident in the small reprojection error of 0.15 pxl.

Finally, we check the extrinsic parameters by using the shifts of camera_1 (see Table 10.12). The results are similar to calibrating two cameras so that

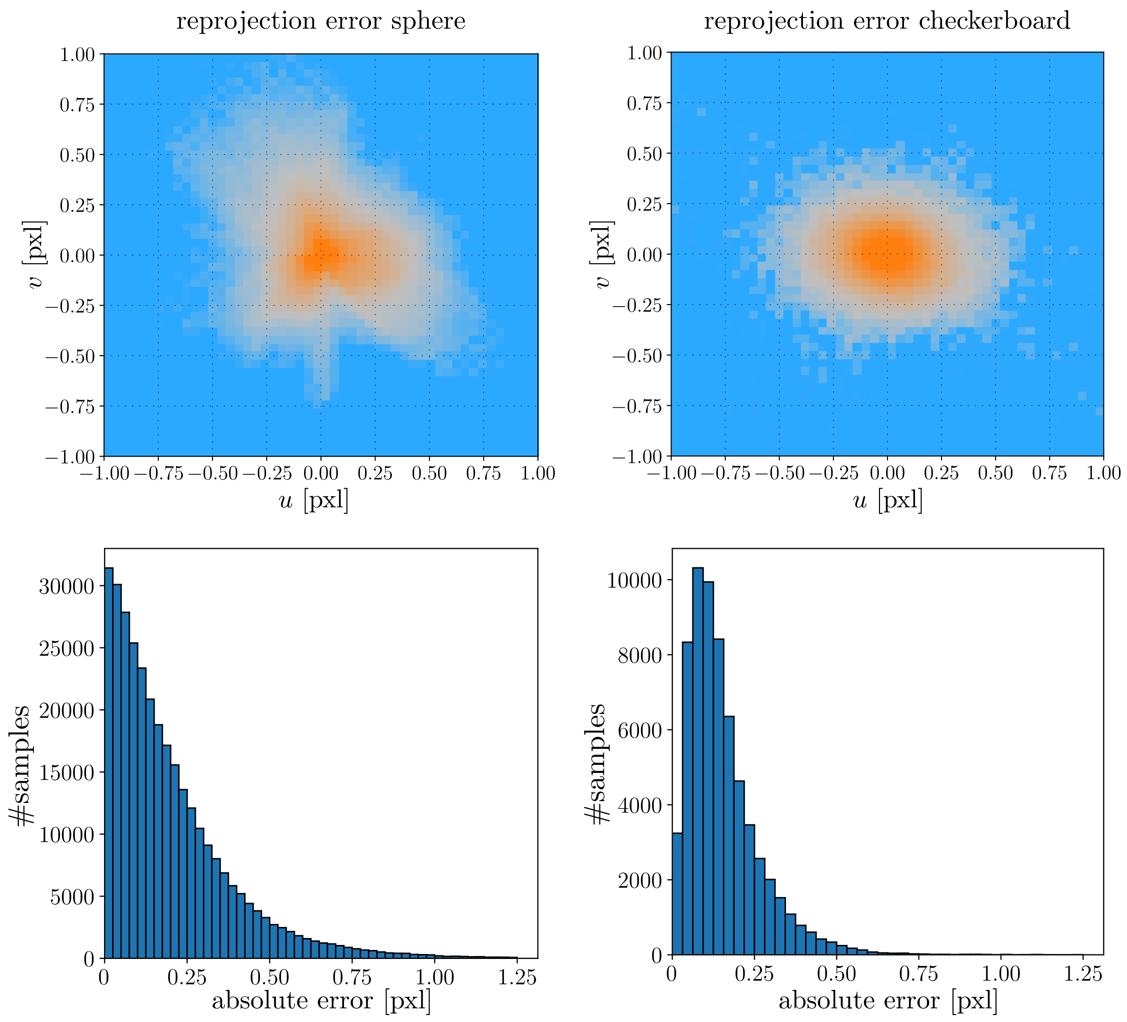


Figure 10.19: Visualization of the reprojection error distributions of one camera. The results at the left are generated by calibrating two cameras and two LiDARs with the joint calibration framework. The distributions at the right result from calibrating two cameras with the reference method. In the top, the reprojection error distribution in u and v directions is plotted. In the bottom row, a histogram over absolute reprojection errors is shown.

we infer again that, when having two cameras, additional data of our LiDARs does not significantly improve the calibration results.

10.5 External Keypoint Calibration

In this section, we evaluate the method for estimating external keypoints on the example of calibrating a sensor setup relative to the rear axle of our experimental vehicle. As explained in chapter 9, additional cameras, that are placed around the car, are used to detect the center points of the wheels by triangulation. Based on the estimated wheel center points, the pose of the rear axle can be calculated.

First, we use simulation to analyze the calibration accuracy for different parameters. Two cameras with a focal length of 1300 pxl are placed in a distance r to each wheel (see Figure 10.20). The two rays that intersect in a wheel center are rotated by an angle α to each other. We simulate these rays by adding zero-mean Gaussian noise with $\sigma_{\text{trans}} = 3\,\text{mm}$ to the origin of the rays and adding zero-mean Gaussian noise with $\sigma_{\text{rot}} = 0.1\,°$ to the direction of the rays to model a calibration error of the sensor poses (values based on results of subsection 10.2.2). Additionally, we add another zero-mean Gaussian angle noise of $\sigma_{\text{pxl}} = 0.04\,°$ which models the detection error of the projected wheel center in the images. For a realistic simulation, we use the dimensions of our experimental vehicle with a wheel base of 2.9 m and an axle width of 1.8 m.
We run multiple experiments with different intersection angles α, different distances r and different scaling factors s_Δ for the calibration error noise in translation and rotation ($\sigma'_{\text{trans}} = s_\Delta \sigma_{\text{trans}}$ and $\sigma'_{\text{rot}} = s_\Delta \sigma_{\text{rot}}$). To quantify the calibration results, the difference of the 2D rear axle pose to the ground truth is calculated. The yaw, longitudinal and lateral errors are denoted as $\Delta\varphi$, Δx and Δy, respectively. The results are given in Figure 10.21. We use the setting with $\alpha = 90\,°$, $r = 2\,\text{m}$ and $s_\Delta = 1$ as our reference parameter set (marked with red).
First, we evaluate different settings for the incidental angle α. The smaller α is set, the smaller the error in yaw and longitudinal direction gets but the higher the error in lateral direction becomes. This can be explained by the increasing range error and decreasing angle error of the estimated center wheel position for decreasing α due to triangulation. The errors in longitudinal and lateral

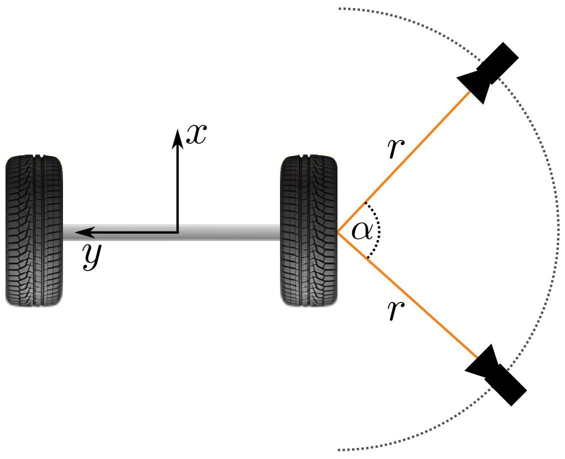


Figure 10.20: Two cameras are placed in front of each wheel to estimate the pose of the rear axle relative to the main sensor setup on a car. The coordinate system of the rear axle has its origin on the center of the rear axle. The x-axis is aligned with the forward driving direction. The rays of the cameras that intersect the wheel center are rotated by an angle α to each other. The cameras are placed in a distance of r to the wheel.

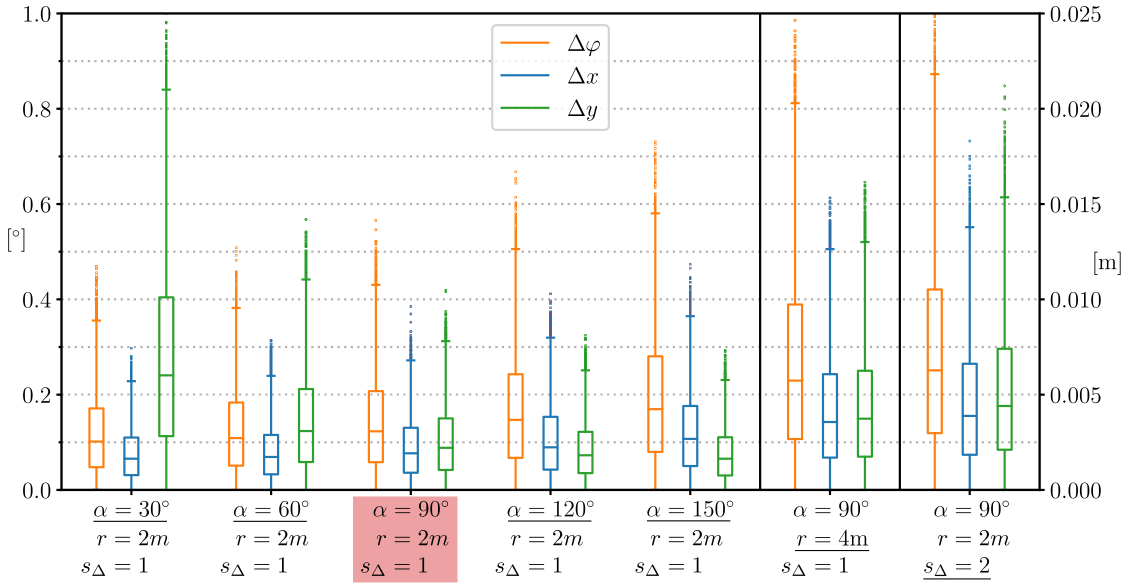


Figure 10.21: Errors of rear axle calibration for different camera positions defined by the distance r to the wheel center and the intersection angle α of the rays from the cameras that intersect in the wheel center. Additionally, the scaling factor s_Δ is varied to scale the calibration error on the camera poses by $\sigma'_{\text{trans}} = s_\Delta \sigma_{\text{trans}}$ and $\sigma'_{\text{rot}} = s_\Delta \sigma_{\text{rot}}$. The reference parameter set is highlighted in red. For all other parameter sets, the changed parameter is underlined [4].

direction are similar for the reference parameter set with $\alpha = 90\,°$.
Next, the distance r of the cameras to the wheel center is doubled. All errors increase significantly. So, to keep the errors small, the cameras should be placed close to the wheels. But if the cameras are placed too close to the wheels, the calibration target cannot be observed ideally anymore. A good compromise is the reference setting of $r = 2\,\text{m}$.
Finally, the scaling factor s_Δ is set to 2 which doubles the noise that models the calibration errors of the camera poses. Doubling the noise roughly doubles the pose error of the rear axle for our experiments. So, the calibration errors of the sensor poses have a major effect on the results. Therefore, it is important to use cameras that can be well calibrated (high resolution sensor and a high quality lens).
Overall, the mean position error of the rear axle is lower than 1 cm in longitudinal and lateral direction and the mean yaw error is lower than $0.3\,°$. The mean accuracy is sufficient for common tasks like e.g. multi sensor localization with cameras, LiDARs and vehicle odometry based on wheel encoders and steering angle measurements (see [7]).

In addition to the experiments in simulation, we use real data to qualitatively assess the calibration to the rear axle. The main sensor setup consists of three LiDARs which are mounted on the roof of our experimental vehicle and a front camera that is mounted behind the windshield. Two external cameras are used to estimate the wheel center positions. The cameras see both wheels on one side. They are repositioned once so that all four wheels are observed. In Figure 10.22, the point cloud data from the LiDARs are visualized in a 3D view. The plotted coordinate systems (red, green, blue axes) represent the estimated poses of the sensors. The CAD model is an accurate model of our car. The rear axle of the CAD model is aligned with the estimated rear axle from the calibration process. The camera appears to be directly behind the windshield which is consistent with the real setup. Further, the LiDAR points that hit the real car lie on the surface of the CAD model. Based on this, we conclude that the rear axle is successfully estimated.

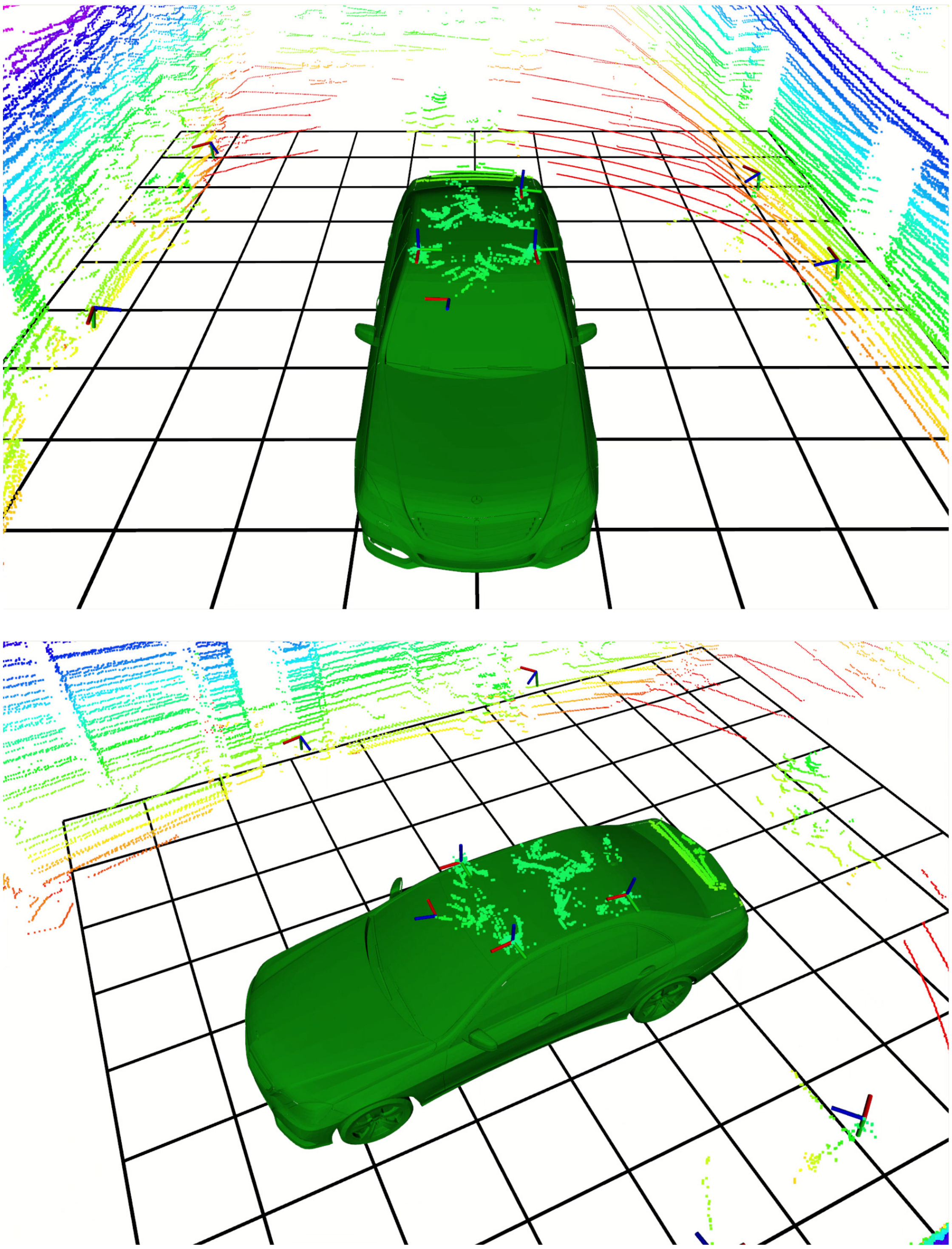


Figure 10.22: Result of calibrating the main sensor setup consisting of three LiDARs on the roof of our experimental vehicle and one camera behind the windshield relative to the rear axle by using two external cameras that are repositioned once. The estimated coordinate systems of the sensors are visualized as red, green and blue axes. The real point cloud data of the LiDARs are shown. Additionally, an accurate CAD model of the car is aligned with the estimated rear axle.

11 Conclusion and Future Work

In this work, we developed novel methods to calibrate multimodal sensor setups consisting of cameras and range sensors such as LiDARs and radars. All methods use a Styrofoam sphere as a calibration target. A corner reflector is mounted in the inside of the hollow sphere to integrate radars in the calibration process.

We developed target detectors for cameras and range sensors that are evaluated in simulation and on real data. For the camera sphere detector, we analyzed error sources such as intensity noise, inhomogeneous lighting and defocus. For the LiDAR sphere detector, the effect of range noise was examined. Further, the distributions of the detections are determined in real data experiments. We found that most components can be well approximated by Gaussian distributions.

We introduced our Euclidean calibration framework which is the simplest of our three multimodal calibration methods. It uses sphere center observations to estimate the poses of the sensors. Depending on the sensor, observations have different geometric types. Cameras generate ray observations or point observations if the actual radius of the sphere is used. Range sensors observe the sphere center as points. To link time-asynchronous observations of different sensors, the sphere position is interpolated linearly. The calibration problem is formulated as a minimization problem of distances of sphere center observations at the same point in time. Depending on the observation types, these are ray-to-ray, ray-to-point or point-to-point distances.
We used our simulation environment to analyze the effect of different error sources. We found that errors due to interpolation of the target positions are negligible. The most severe errors result from inaccurate timestamps of the sensor data if no proper time management system is installed. On real data experiments with a camera and a LiDAR, we found an absolute mean translation error of less than 3 mm and a rotation error of less than 0.1 °, which compares

well to state of the art methods.

In our Euclidean calibration framework, all observations influence the calibration result the same. We introduced our probabilistic calibration framework that, additionally, takes the information about observation uncertainties into account by using a probabilistic formulation of the calibration problem. More precisely, the joint probability density function of the sensor poses and the target positions, given the target observations, is maximized. By using a generic sensor model that is parameterized for each sensor individually, the different observation characteristics are encoded. Under certain assumptions on the observation distributions, the calibration problem can be formulated as a weighted least-squares problem that can be efficiently solved.
On simulated and real data, we showed that the probabilistic calibration framework significantly outperforms the Euclidean calibration framework. The downside of incorporating the observation characteristics into the calibration problem is that, for each sensor, the observation noise has to be estimated. This can be a time consuming process. We showed for different sensor setups that, even for rough noise estimates, the probabilistic approach outperforms the Euclidean calibration framework. This is an essential finding for the practical use of the method.

Both, the Euclidean and the probabilistic calibration framework estimate the poses of the sensors. We introduced our joint calibration framework that estimates intrinsic parameters in addition to the sensor poses. Again, the calibration is formulated in a probabilistic manner. Raw measurements instead of precalculated sphere center observations are incorporated into the problem by using individual measurement models. Determining the noise characteristics of the measurements is often trivial since the datasheets of the sensors provide this information.
On simulated data, we showed that, for a setup that consists of a single camera and a LiDAR, the estimation of the camera intrinsics significantly improves by adding the range measurements from the LiDAR to the calibration problem. We reproduced this result on real data. Further, we compared our method to a state of the art reference method that uses checkerboards. The reference method has a significantly higher repeatability than our method in the case of calibrating a single camera. By adding LiDARs, our method outperforms the reference method for a single camera with respect to repeatability of the intrinsic camera parameters. For a setup of two cameras, our and the reference

method showed comparable repeatability for the intrinsics. We further showed, by means of known movement of one camera, that the relative positions of the cameras to each other are estimated with sub-millimeter accuracy for our and the reference method.

An extension of the calibration methods is to calibrate the sensors relative to keypoints e.g. a car's rear axle. The main idea is to temporarily extend the main setup by sensors that observe the keypoints. We evaluated the extension qualitatively on real data and quantitatively in simulation. We concluded that the method is a simple and effective alternative to e.g. motion-based rear axle calibration. Additionally, it is a very flexible approach that can further be used e.g. for estimating the position of GNSS antennas.

There are several promising directions to extend this work:
First, a logical next step would be to incorporate full LiDAR intrinsics into the calibration process. In our experience, the intrinsic parameters of LiDARs have to be recalibrated over time. Sending the sensor back to the manufacturer is costly and time consuming. We propose to calibrate LiDAR intrinsics in a joint calibration problem with other sensors such as cameras. The intrinsic parameters can profit significantly from additional information of other sensors. The problem is well posed for a sensor setup consisting of a LiDAR and a camera if the radius of the sphere is known. A simple alternative is to use an already intrinsically well calibrated LiDAR to calibrate the intrinsic parameters of another LiDAR.
Another extension would be to allow multiple spheres as calibration targets. This will reduce the recording time to a few seconds if the targets are spread suitably. The goal would be to equip a garage with e.g. five spheres so that a single drive into the garage provides enough measurements to calibrate the sensor setup, at least its extrinsic parameters.
Another idea is to extend the calibration target for even better calibration results. Especially promising would be to combine a checkerboard with our spherical calibration target. The corners of a checkerboard pattern can be detected very accurately by a camera but has disadvantages for calibrating range sensors which can be compensated by fixing our spherical target to the board. We expect additional improvement for all parameters of all sensors.

A Appendix

A.1 Calculation of camera ray direction with projection model

The goal is to find an explicit formula for the ray direction $\boldsymbol{d}$ in terms of the normed undistorted image point coordinates (u_d, v_d). We use a projection model $\mathcal{A}(\theta) = r_d$ that defines the relation between inclination angle θ and normed image point radius $r_d = \sqrt{u_d^2 + v_d^2}$. Independent of the projection model, we have

$$\boldsymbol{d} = \begin{pmatrix} \sin(\theta)\cos(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\theta) \end{pmatrix} , \tag{A.1}$$

with the azimuthal angle $\varphi = \arctan(v_d/u_d)$. We reformulate $\sin(\varphi)$ and $\cos(\varphi)$ to

$$\sin(\varphi) = \sin\left[\arctan\left(\frac{v_d}{u_d}\right)\right] = \frac{\left(\frac{v_d}{u_d}\right)}{\sqrt{1+\left(\frac{v_d}{u_d}\right)^2}} = \frac{v_d}{r_d} \tag{A.2}$$

$$\cos(\varphi) = \cos\left[\arctan\left(\frac{v_d}{u_d}\right)\right] = \frac{1}{\sqrt{1+\left(\frac{v_d}{u_d}\right)^2}} = \frac{u_d}{r_d} \quad . \tag{A.3}$$

For each of the four projection models presented in Equation 3.4-3.7, we provide the final results for the direction $\boldsymbol{d}$ of the 3D ray:

Projective:

$$\mathcal{A}(\theta) = \tan(\theta) = r_d \tag{A.4}$$

$$\theta = \arctan(r_d) \tag{A.5}$$

$$\sin(\theta) = \sin[\arctan(r_d)] = \frac{r_d}{\sqrt{1 + r_d^2}} \tag{A.6}$$

$$\cos(\theta) = \cos[\arctan(r_d)] = \frac{1}{\sqrt{1 + r_d^2}} \tag{A.7}$$

$$\boldsymbol{d} = \begin{pmatrix} \sin(\theta)\cos(\varphi) \\ \sin(\theta)\sin(\varphi) \\ \cos(\theta) \end{pmatrix} = \begin{pmatrix} \frac{r_d}{\sqrt{1+r_d^2}} \frac{u_d}{r_d} \\ \frac{r_d}{\sqrt{1+r_d^2}} \frac{v_d}{r_d} \\ \frac{1}{\sqrt{1+r_d^2}} \end{pmatrix} = \begin{pmatrix} \frac{u_d}{\sqrt{1+r_d^2}} \\ \frac{v_d}{\sqrt{1+r_d^2}} \\ \frac{1}{\sqrt{1+r_d^2}} \end{pmatrix} \tag{A.8}$$

Stereographic:

$$\mathcal{A}(\theta) = 2\tan\left(\frac{\theta}{2}\right) = r_d \tag{A.9}$$

$$\theta = 2\arctan\left(\frac{r_d}{2}\right) \tag{A.10}$$

$$\begin{aligned} \sin(\theta) &= \sin\left[2\arctan\left(\frac{r_d}{2}\right)\right] = \\ &= 2\sin\left[\arctan\left(\frac{r_d}{2}\right)\right]\cos\left[\arctan\left(\frac{r_d}{2}\right)\right] \\ &= 2\frac{\frac{r_d}{2}}{\sqrt{1+\left(\frac{r_d}{2}\right)^2}}\frac{1}{\sqrt{1+\left(\frac{r_d}{2}\right)^2}} = \frac{4r_d}{4+r_d^2} \end{aligned} \tag{A.11}$$

$$\begin{aligned}\cos(\theta) &= \cos\left[2\arctan\left(\frac{r_d}{2}\right)\right]\\ &= \cos^2\left[\arctan\left(\frac{r_d}{2}\right)\right] - \sin^2\left[\arctan\left(\frac{r_d}{2}\right)\right]\\ &= \frac{4}{4+r_d^2} - \frac{r_d^2}{4+r_d^2} = \frac{4-r_d^2}{4+r_d^2}\end{aligned} \tag{A.12}$$

$$\boldsymbol{d} = \begin{pmatrix}\sin(\theta)\cos(\varphi)\\ \sin(\theta)\sin(\varphi)\\ \cos(\theta)\end{pmatrix} = \begin{pmatrix}\frac{4r_d}{4+r_d^2}\frac{u_d}{r_d}\\ \frac{4r_d}{4+r_d^2}\frac{v_d}{r_d}\\ \frac{4-r_d^2}{4+r_d^2}\end{pmatrix} = \begin{pmatrix}\frac{4u_d}{4+r_d^2}\\ \frac{4v_d}{4+r_d^2}\\ \frac{4-r_d^2}{4+r_d^2}\end{pmatrix} \tag{A.13}$$

Equiangular:

$$\mathcal{A}(\theta) = \theta \tag{A.14}$$

$$\theta = r_d \tag{A.15}$$

$$\boldsymbol{d} = \begin{pmatrix}\sin(\theta)\cos(\varphi)\\ \sin(\theta)\sin(\varphi)\\ \cos(\theta)\end{pmatrix} = \begin{pmatrix}\operatorname{sinc}(r_d)u_d\\ \operatorname{sinc}(r_d)v_d\\ \cos(r_d)\end{pmatrix} \tag{A.16}$$

Equisolid:

$$\mathcal{A}(\theta) = 2\sin\left(\frac{\theta}{2}\right) = r_d \tag{A.17}$$

$$\theta = 2\arcsin\left(\frac{r_d}{2}\right) \tag{A.18}$$

$$\begin{aligned}\sin(\theta) &= \sin\left[2\arcsin\left(\frac{r_d}{2}\right)\right]\\ &= 2\sin\left[\arcsin\left(\frac{r_d}{2}\right)\right]\cos\left[\arcsin\left(\frac{r_d}{2}\right)\right]\\ &= r_d\sqrt{1-\left(\frac{r_d}{2}\right)^2}\end{aligned} \tag{A.19}$$

$$\begin{aligned}\cos(\theta) &= \cos\left[2\arcsin\left(\frac{r_d}{2}\right)\right]\\ &= \cos^2\left[\arcsin\left(\frac{r_d}{2}\right)\right] - \sin^2\left[\arcsin\left(\frac{r_d}{2}\right)\right]\\ &= 1 - \left(\frac{r_d}{2}\right)^2 - \left(\frac{r_d}{2}\right)^2 = 1 - \frac{r_d^2}{2}\end{aligned} \tag{A.20}$$

$$\boldsymbol{d} = \begin{pmatrix}\sin(\theta)\cos(\varphi)\\ \sin(\theta)\sin(\varphi)\\ \cos(\theta)\end{pmatrix} = \begin{pmatrix}\sqrt{1-\left(\frac{r_d}{2}\right)^2}u_d\\ \sqrt{1-\left(\frac{r_d}{2}\right)^2}v_d\\ 1-\frac{r_d^2}{2}\end{pmatrix} \tag{A.21}$$

A.2 Calculation of tangent angle and its variance

The goal is to calculate the tangent at an edge point based on edge points in a small neighborhood. Further, the variance of the tangent angle shall be derived from the measurement noise of the edge point positions.
As explained in subsection 3.3.3, PCA can be used as a tool to find the best line fit with orthogonal distance measure. For PCA, we need to calculate the sample covariance matrix of all n points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in the neighborhood. Actually, we can drop the scaling factor $(n-1)$ since it does not influence the tangent direction. We use the following notation $\boldsymbol{x}_i = (x_i, y_i)^T$ and define the scaled sample covariance matrix as

$$\boldsymbol{A} = \begin{pmatrix}a & b\\ c & d\end{pmatrix} = \begin{pmatrix}\sum(x_i-m_x)(x_i-m_x) & \sum(x_i-m_x)(y_i-m_y)\\ \sum(x_i-m_x)(y_i-m_y) & \sum(y_i-m_y)(y_i-m_y)\end{pmatrix} \quad , \tag{A.22}$$

with the mean point

$$\boldsymbol{m} = \begin{pmatrix}m_x\\ m_y\end{pmatrix} = \frac{1}{n}\begin{pmatrix}\sum x_i\\ \sum y_i\end{pmatrix} \quad . \tag{A.23}$$

The eigenvector of $\boldsymbol{A}$ with the largest eigenvalue can be proven to be

$$\boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{cases} \begin{pmatrix} L-d \\ c \end{pmatrix} & \text{if } b \neq 0 \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \text{if } b = 0 \text{ and } a > c \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} & \text{if } b = 0 \text{ and } c \geq a \end{cases} , \tag{A.24}$$

with $L = \frac{a+d}{2} + \sqrt{\frac{(a+d)^2}{4} - ad + bc}$. The largest eigenvector $\boldsymbol{v}$ represents the direction of the line fit. We use the additional convention that the sphere projection is to the right of the tangent so that the sign of the tangent direction has to be adjusted accordingly. The tangent angle can be calculated by using the *2-argument arcus tangent*

$$\varphi = \arctan2(v_2, v_1) \ . \tag{A.25}$$

At this point, we estimated the tangent direction and need to derive the variance of the tangent angle. We assume Gaussian noise orthogonal to the tangent direction for the points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$. Since the eigenvector $\boldsymbol{v}$ is a non-linear function in the points, we use a linear approximation to derive the variance of the tangent angle by the error propagation concept explained in subsection 3.3.4. For this, we need to calculate the Jacobian

$$\boldsymbol{J} = \nabla\varphi^T(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \ . \tag{A.26}$$

First, we simplify $\boldsymbol{v}$ by only considering the case $b \neq 0$. If $b = 0$, we simply rotate all the points around the average point $\boldsymbol{m}$ by e.g. $45\,^\circ$ to have $b \neq 0$. The variance is the same for the rotated case. We use the chain rule to find

$$\nabla_{\boldsymbol{x}_i}\varphi^T(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) = \begin{pmatrix} \frac{\partial\varphi}{\partial x_i} & \frac{\partial\varphi}{\partial y_i} \end{pmatrix} = \begin{pmatrix} \frac{\partial\varphi}{\partial\left(\frac{v_2}{v_1}\right)} \frac{\partial\left(\frac{v_2}{v_1}\right)}{\partial x_i} & \frac{\partial\varphi}{\partial\left(\frac{v_2}{v_1}\right)} \frac{\partial\left(\frac{v_2}{v_1}\right)}{\partial y_i} \end{pmatrix} \ . \tag{A.27}$$

We have

$$\frac{\partial\varphi}{\partial\left(\frac{v_2}{v_1}\right)} = \frac{\partial \mathrm{arctan2}(v_2, v_1)}{\partial\left(\frac{v_2}{v_1}\right)} \tag{A.28}$$

$$= \frac{\partial \arctan\left(\frac{v_2}{v_1}\right)}{\partial\left(\frac{v_2}{v_1}\right)} \tag{A.29}$$

$$= \frac{1}{1+\left(\frac{v_2}{v_1}\right)^2} \tag{A.30}$$

$$= \frac{(L-d)^2}{(L-d)^2+c^2} \quad . \tag{A.31}$$

We further have

$$\frac{\partial\frac{v_2}{v_1}}{\partial x_i} = \frac{\partial\frac{c}{L-d}}{\partial x_i} = \frac{\partial c}{\partial x_i}\frac{1}{L-d} + c\frac{\partial(L-d)^{-1}}{\partial x_i} \tag{A.32}$$

$$= \frac{\partial c}{\partial x_i}\frac{1}{L-d} - \frac{c\left(\frac{\partial L}{\partial x_i} - \frac{\partial d}{\partial x_i}\right)}{(L-d)^2} \tag{A.33}$$

$$= \frac{\frac{\partial c}{\partial x_i}(L-d) - c\left(\frac{\partial L}{\partial x_i} - \frac{\partial d}{\partial x_i}\right)}{(L-d)^2} \tag{A.34}$$

and

$$\frac{\partial\frac{v_2}{v_1}}{\partial y_i} = \frac{\partial\frac{c}{L-d}}{\partial y_i} = \frac{\partial c}{\partial y_i}\frac{1}{L-d} + c\frac{\partial(L-d)^{-1}}{y_i} \tag{A.35}$$

$$= \frac{\partial c}{\partial y_i}\frac{1}{L-d} - \frac{c\left(\frac{\partial L}{\partial y_i} - \frac{\partial d}{\partial y_i}\right)}{(L-d)^2} \tag{A.36}$$

$$= \frac{\frac{\partial c}{\partial y_i}(L-d) - c\left(\frac{\partial L}{\partial y_i} - \frac{\partial d}{\partial y_i}\right)}{(L-d)^2} \quad . \tag{A.37}$$

We combine Equations A.27-A.37 in

$$\nabla_{\boldsymbol{x}_i}\varphi^T(\boldsymbol{x}_1,\dots,\boldsymbol{x}_n) = \begin{pmatrix} \frac{\frac{\partial c}{\partial x_i}(L-d)-c\left(\frac{\partial L}{\partial x_i}-\frac{\partial d}{\partial x_i}\right)}{(L-d)^2+c^2} & \frac{\frac{\partial c}{\partial y_i}(L-d)-c\left(\frac{\partial L}{\partial y_i}-\frac{\partial d}{\partial y_i}\right)}{(L-d)^2+c^2} \end{pmatrix}. \tag{A.38}$$

Next, we determine

$$\frac{\partial L}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\frac{a+d}{2}+\sqrt{\frac{(a+d)^2}{4}-ad+bc}\right) \tag{A.39}$$

$$= \frac{\frac{\partial a}{\partial x_i}+\frac{\partial d}{\partial x_i}}{2} + \frac{1}{2\sqrt{\frac{(a+d)^2}{4}-ad+bc}} \left(\frac{(a+d)\left(\frac{\partial a}{\partial x_i}+\frac{\partial d}{\partial x_i}\right)}{2} - \frac{\partial a}{\partial x_i}d - a\frac{\partial d}{\partial x_i} + \frac{\partial b}{\partial x_i}c + b\frac{\partial c}{\partial x_i}\right) \tag{A.40}$$

and

$$\frac{\partial L}{\partial y_i} = \frac{\partial}{\partial y_i}\left(\frac{a+d}{2}+\sqrt{\frac{(a+d)^2}{4}-ad+bc}\right) \tag{A.41}$$

$$= \frac{\frac{\partial a}{\partial y_i}+\frac{\partial d}{\partial y_i}}{2} + \frac{1}{2\sqrt{\frac{(a+d)^2}{4}-ad+bc}} \left(\frac{(a+d)\left(\frac{\partial a}{\partial y_i}+\frac{\partial d}{\partial y_i}\right)}{2} - \frac{\partial a}{\partial y_i}d - a\frac{\partial d}{\partial y_i} + \frac{\partial b}{\partial y_i}c + b\frac{\partial c}{\partial y_i}\right). \tag{A.42}$$

Finally, we calculate

$$\frac{\partial a}{\partial x_i} = \frac{\partial \sum_j \left(x_j - \frac{1}{n}\sum_k x_k\right)^2}{\partial x_i} \tag{A.43}$$

$$= \sum_{j \neq i} \left[2\left(x_j - \frac{1}{n}\sum_k x_k\right)\left(-\frac{1}{n}\right)\right] + 2\left(x_i - \frac{1}{n}\sum_k x_k\right)\left(1 - \frac{1}{n}\right) \tag{A.44}$$

$$= \sum_{j \neq i} \left[2\left(x_j - m_x\right)\left(-\frac{1}{n}\right)\right] + 2\left(x_i - m_x\right)\left(1 - \frac{1}{n}\right) \tag{A.45}$$

$$= -\frac{2}{n}\sum_j \left[2\left(x_j - m_x\right)\right] + 2(x_i - m_x) \tag{A.46}$$

$$= -\frac{2}{n}\sum_j x_j + \frac{2}{n}\sum_j m_x + 2(x_i - m_x) \tag{A.47}$$

$$= 2(x_i - m_x) \tag{A.48}$$

$$\frac{\partial b}{\partial x_i} = \frac{\partial \sum_j \left[\left(x_j - \frac{1}{n}\sum_k x_k\right)\left(y_j - \frac{1}{n}\sum_k y_k\right)\right]}{\partial x_i} \tag{A.49}$$

$$= \sum_{j \neq i} \left[-\frac{1}{n}\left(y_j - m_y\right)\right] + \left(1 - \frac{1}{n}\right)\left(y_i - m_y\right) \tag{A.50}$$

$$= \frac{(n-1)y_i - \sum_{j \neq i} y_j}{n} \tag{A.51}$$

$$= y_i - m_y \tag{A.52}$$

$$\frac{\partial c}{\partial x_i} = \frac{\partial b}{\partial x_i} = y_i - m_y \tag{A.53}$$

$$\frac{\partial d}{\partial x_i} = \frac{\partial \sum_j \left[\left(y_j - \frac{1}{n}\sum_k y_k\right)\left(y_j - \frac{1}{n}\sum_k y_k\right)\right]}{\partial x_i} \tag{A.54}$$

$$= 0 \tag{A.55}$$

and the same calculations are performed to find

$$\frac{\partial a}{\partial y_i} = 0 \tag{A.56}$$

$$\frac{\partial b}{\partial y_i} = x_i - m_x \tag{A.57}$$

$$\frac{\partial c}{\partial y_i} = x_i - m_x \tag{A.58}$$

$$\frac{\partial d}{\partial y_i} = 2(y_i - m_y) \quad . \tag{A.59}$$

From all these equations, we can calculate the Jacobian $\boldsymbol{J} = \nabla\varphi^T(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) = \left(\nabla_{\boldsymbol{x}_1}\varphi^T(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \; ... \; \nabla_{\boldsymbol{x}_n}\varphi^T(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)\right)$. Since the noise of the edge points is assumed to be orthogonal to the edge tangent, we need the directional derivatives:

$$\nabla^{\perp}_{\boldsymbol{x}_i}\varphi^T(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) = \nabla_{\boldsymbol{x}_i}\varphi^T(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \cdot \mathbf{n} \quad , \tag{A.60}$$

with the normal $\mathbf{n}$. The final variance of the tangential angle can be calculated by

$$\sigma_\varphi^2 = \|\nabla^{\perp}\varphi(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)\|^2 \sigma_{\text{pix}}^2 \tag{A.61}$$

$$= \| \left(\nabla^{\perp}_{\boldsymbol{x}_1}\varphi(\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \; ... \; \nabla^{\perp}_{\boldsymbol{x}_n}\varphi(\boldsymbol{x}_1, ..., \boldsymbol{x}_n)\right) \|^2 \sigma_{\text{pix}}^2 \quad . \tag{A.62}$$

Bibliography

[ABB12] Hatem Alismail, L. Douglas Baker, and Brett Browning. Automatic calibration of a range sensor and camera system. In *IEEE International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 286–292, Zurich, Switzerland, 2012.

[AD03] Motilal Agrawal and Larry S. Davis. Camera calibration using spheres: A semi-definite programming approach. In *IEEE International Conference on Computer Vision (ICCV)*, pages 782–789, Nice, France, 2003.

[AKU^{+}06] Alen Alempijevic, Sarath Kodagoda, James Underwood, Suresh Kumar, and Gamini Dissanayake. Mutual information based sensor registration and calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 25–30, Beijing, China, 2006.

[Arr98] Kai O. Arras. *An Introduction To Error Propagation: Derivation, Meaning and Examples of Equation Cy= Fx Cx FxT*. Technical Report, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1998. Published online by ETH Zurich. `https://doi.org/10.3929/ethz-a-010113668`. Accessed: 2019-11-15.

[BHL^{+}05] Ilja N. Bronstein, Juraj Hromkovic, Bernd Luderer, Hans-Rudolf Schwarz, Jochen Blath, Alexander Schied, Stephan Dempe, Gert Wanka, and Siegfried Gottwald. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, Germany, 6. edition, 2005.

[BHWM06] Klemens Burg, Herbert Haf, Friedrich Wille, and Andreas Meister. *Höhere Mathematik für Ingenieure, Band I*. Teubner, Wiesbaden, Germany, 7. edition, 2006.

[Bil09] Stanley Bileschi. Fully automatic calibration of lidar and video streams from a vehicle. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 1457–1464, Kyoto, Japan, 2009.

[Bis06] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer Science+Business Media, New York, USA, 1. edition, 2006.

[Bjö96] Åke Björck. *Numerical methods for least squares problems*. SIAM, Philadelphia, USA, 1. edition, 1996.

[Bla10] Jose-Luis Blanco. *A tutorial on se(3) transformation parameterizations and on-manifold optimization*. Technical Report, University of Malaga, Malaga, Spain, 2010.

[BM12] Saket Bhardwaj and Ajay Mittal. A survey on various edge detector techniques. *Procedia Technology*, 4:220–226, 2012.

[BMRS12] Luigi Barazzetti, Luigi Mussio, Fabio Remondino, and Marco Scaioni. Targetless camera calibration. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, XXXVIII-5/W16:335–342, 2012.

[BPDA00] Faysal Boughorbal, David L. Page, Christophe Dumont, and Mongi A. Abidi. Registration and integration of multisensor data for photorealistic scene reconstruction. In *AIPR Workshop: 3D Visualization for Data Exploration and Decision Making*, pages 74 – 84, Washington, DC, USA, 2000.

[BS18] Johannes Beck and Christoph Stiller. Generalized b-spline camera model. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 2137–2142, Suzhou, China, 2018.

[Bus03] Samuel R. Buss. *3D computer graphics: a mathematical introduction with OpenGL*. Cambridge University Press, Cambridge, UK, 1. edition, 2003.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 8(6):679–698, 1986.

[CKB16] Juan Castorena, Ulugbek S. Kamilov, and Petros T. Boufounos. Autocalibration of lidar and optical cameras via edge alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2862–2866, Shanghai, China, 2016.

[Cox72] Maurice G. Cox. The numerical evaluation of b-splines. *IMA Journal of Applied Mathematics*, 10(2):134–149, 1972.

[Dav75] Larry S. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4(3):248–270, 1975.

[DB72] Carl De Boor. On calculating with b-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

[DB78] Carl De Boor. *A practical guide to splines*. Springer, New York, USA, 1. edition, 1978.

[DCRK17] Ankit Dhall, Kunal Chelani, Vishnu Radhakrishnan, and K. Madhava Krishna. Lidar-camera calibration using 3d-3d point correspondences. *arXiv preprint arXiv:1705.09785*, 2017.

[DDL94] Nadine Daucher, Michel Dhome, and Jean-Thierry Lapresté. Camera calibration from spheres images. In *European Conference on Computer Vision (ECCV)*, pages 447–454, Stockholm, Sweden, 1994.

[DH71] Richard O. Duda and Peter E. Hart. *Use of the hough transformation to detect lines and curves in pictures*. Technical Report, Sri International Artificial Intelligence Center, Menlo Park, USA, 1. edition, 1971.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern recognition and scene analysis*. Wiley, New York, USA, 1. edition, 1973.

[DHS09] Thao Dang, Christian Hoffmann, and Christoph Stiller. Continuous stereo self-calibration by camera parameter tracking. *IEEE Transactions on Image Processing*, 18(7):1536–1550, 2009.

[DKG19] Joris Domhof, Julian F. P. Kooij, and Dariu M. Gavrila. An extrinsic calibration tool for radar, camera and lidar. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8107–8113, Montreal, Canada, 2019.

[DKL98] Erik B. Dam, Martin Koch, and Martin Lillholm. *Quaternions, interpolation and animation*. Technical Report DIKU-TR-98/5, University of Copenhagen, Copenhagen, Denmark, 1998.

[DP73] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[EG04] Jerry Eriksson and Mårten Gulliksson. Local results for the gauss-newton method on constrained rank-deficient nonlinear least squares. *Mathematics of Computation*, 73(248):1865–1883, 2004.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Fia09] Mark Fiala. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(7):1317–1324, 2009.

[Fis95] Nicholas I. Fisher. *Statistical analysis of circular data*. Cambridge University Press, Cambridge, UK, 1. edition, 1995.

[GJMSMCMJ14] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco J. Madrid-Cuevas, and Manuel J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[GLL13] Xiaojin Gong, Ying Lin, and Jilin Liu. 3d lidar-camera extrinsic calibration using an arbitrary trihedron. *Sensors*, 13(2):1902–1918, 2013.

[GMCS12] Andreas Geiger, Frank Moosmann, Ömer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943, St. Paul, USA, 2012.

[Hei00] Janne Heikkila. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(10):1066–1077, 2000.

[HMES16] M. Hassanein, A. Moussa, and N. El-Sheimy. A new automatic system calibration of multi-cameras and lidar sensors. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41:589–594, 2016.

[Hou62] Paul V. C. Hough. Method and means for recognizing complex patterns, 1962. US Patent 3,069,654.

[HWL83] Robert M. Haralick, Layne T. Watson, and Thomas J. Laffey. The topographic primal sketch. *The International Journal of Robotics Research*, 2(1):50–72, 1983.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer visionfusing*. Cambridge University Press, New York, USA, 1. edition, 2003.

[IK87] John Illingworth and Josef Kittler. The adaptive hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 9(5):690–698, 1987.

[IOI18] Ryoichi Ishikawa, Takeshi Oishi, and Katsushi Ikeuchi. Lidar and camera calibration using motions estimated by sensor fusion odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7342–7349, Madrid, Spain, 2018.

[JGAK07] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reactable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the International Conference on Tangible and Embedded Interaction*, pages 139–146, Baton Rouge, USA, 2007.

[JKB16] Norman L. Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions*, volume 1 of *Wiley Series in Probability and Statistics*. John Wiley & Sons, New York, USA, 1. edition, 2016.

[Jol02] Ian T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, USA, 2. edition, 2002.

[JS09] Mamta Juneja and Parvinder S. Sandhu. Performance evaluation of edge detection techniques for images in spatial domain. *International Journal of Computer Theory and Engineering (IJCTE)*, 1(5):614, 2009.

[JW02] Friedrich K. Jondral and Anne Wiesler. *Wahrscheinlichkeitsrechnung und stochastische Prozesse: Grundlagen für Ingenieure und Naturwissenschaftler*. Vieweg+Teubner Verlag, Wiesbaden, Germany, 2. edition, 2002.

[Kat] Hirokazu Kato. ARToolKit. `http://www.hitl.washington.edu/artoolkit/`. Accessed: 2019-11-15.

[KB06] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(8):1335–1340, 2006.

[KP10] Abdallah Kassir and Thierry Peynot. Reliable automatic camera-laser calibration. In *Australasian Conference on Robotics & Automation*, pages 1–10, Brisbane, Australia, 2010.

[LDLP19] Johann Laconte, Simon-Pierre Deschênes, Mathieu Labussière, and François Pomerleau. Lidar measurement bias estimation via return waveform modelling in a context of 3d mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8100–8106, Montreal, Canada, 2019.

[LKA17] Jake Lever, Martin Krzywinski, and Naomi S. Altman. Points of significance: Principal component analysis. *Nature Methods*, 14(7):641–642, 2017.

[LLD$^+$07] Ganhua Li, Yunhui Liu, Li Dong, Xuanping Cai, and Dongxiang Zhou. An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3854–3859, San Diego, USA, 2007.

[LT13] Jesse Levinson and Sebastian Thrun. Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems (RSS)*, volume 2, pages 1–8, Berlin, Germany, 2013.

[MKF09] A. Mastin, J. Kepner, and J. Fisher. Automatic registration of lidar and optical images of urban scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2639–2646, Miami, Florida, 2009.

[MW16] Georg R. Mueller and Hans-Joachim Wuensche. Continuous extrinsic online calibration for stereo cameras. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 966–971, Gothenburg, Sweden, 2016.

[NAAR+15] Ghina E. Natour, Omar Ait-Aider, Raphael Rouveure, François Berry, and Patrice Faure. Toward 3d reconstruction of outdoor scenes using an mmw radar and a monocular vision sensor. *Sensors*, 15(10):25937–25967, 2015.

[NF02] Leonid Naimark and Eric Foxlin. Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 27–36, Darmstadt, Germany, 2002.

[NKB19] B. Nagy, L. Kovács, and C. Benedek. Online targetless end-to-end camera-lidar self-calibration. In *International Conference on Machine Vision Applications (MVA)*, pages 1–6, Tokyo, Japan, 2019.

[NW06] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, New York, USA, 2. edition, 2006.

[ON15] Mohammad Omidalizarandi and Ingo Neumann. Comparison of target- and mutual informaton based calibration of terrestrial laser scanner and digital camera for deformation monitoring. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, XL-1-W5:559–564, 2015.

[Ots79] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[PBP13] Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. *Bézier and b-spline techniques*. Springer Science & Business Media, Heidelberg, Germany, 1. edition, 2013.

[PEH18] Zoltán Pusztai, Iván Eichhardt, and Levente Hajder. Accurate calibration of multi-lidar-multi-camera systems. *Sensors*, 18(7):2139, 2018.

[PMP19] Juraj Peršić, Ivan Marković, and Ivan Petrović. Extrinsic 6dof calibration of a radar–lidar–camera system enhanced by radar cross section estimates evaluation. *Robotics and Autonomous Systems*, 114:217–230, 2019.

[PMSE12] Gaurav Pandey, James R. McBride, Silvio Savarese, and Ryan M. Eustice. Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information. In *AAAI Conference on Artificial Intelligence*, pages 2053–2059, Toronto, Canada, 2012.

[Pre70] Judith M. S. Prewitt. Object enhancement and extraction. *Picture Processing and Psychopictorics*, 10(1):15–19, 1970.

[PT97] Les Piegl and Wayne Tiller. *The NURBS book*. Springer, Berlin, Germany, 2. edition, 1997.

[PYW$^+$14] Yoonsu Park, Seokmin Yun, Chee Won, Kyungeun Cho, Kyhyun Um, and Sungdae Sim. Calibration between color camera and 3d lidar instruments with a polygonal planar board. *Sensors*, 14(3):5333–5353, 2014.

[RFFB08] Sergio A. Rodriguez F., Vincent Fremondt, and Philippe Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 214–219, Seoul, Korea, 2008.

[RG04] Michael Rohs and Beat Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. In *Advances in Pervasive Computing*, pages 265–271, Vienna, Austria, 2004.

[RKBL17] Eike Rehder, Christian Kinzig, Philipp Bender, and Martin Lauer. Online stereo camera calibration from scratch. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1694–1699, Los Angeles, USA, 2017.

[RL03] Peter J. Rousseeuw and Annick M. Leroy. *Robust regression and outlier detection*. John Wiley & Sons, Hoboken, USA, 1. edition, 2003.

[Rou84] Peter J. Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

[Rub07] Miguel Rubi. The causality principle: complexity is the limit. In *CAS Confluence. Interdisciplinary Communications*, pages 119–122, Oslo, Norway, 2007.

[RVD06] Peter J. Rousseeuw and Katrien Van Driessen. Computing lts regression for large data sets. *Data Mining and Knowledge Discovery*, 12(1):29–45, 2006.

[SA85] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics and Image Processing*, 30(1):32–46, 1985.

[SA01] Ioannis Stamos and P. K. Alien. Automatic registration of 2-d with 3-d imagery in urban environments. In *IEEE International Conference on Computer Vision (ICCV)*, pages 731 – 736, Vancouver, Canada, 2001.

[SHS07] Davide Scaramuzza, Ahad Harati, and Roland 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)*, pages 4164–4169, San Diego, USA, 2007.

[SMS06] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *IEEE International Conference on Computer Vision Systems (ICVS)*, pages 45–52, New York, USA, 2006.

[Ste98] Carsten Steger. An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(2):113–125, 1998.

[Str15] Tobias Strauss. *Kalibrierung von Multi-Kamera-Systemen - Kombinierte Schätzung von intrinsischem Abbildungsverhalten der einzelnen Kameras und deren relativer Lage zueinander ohne Erfordernis sich überlappender Sichtbereiche*. PhD thesis, Karlsruhe Institute of Technology, KIT Scientific Publishing, Karlsruhe, Germany, 2015.

[STTO04] Shigeki Sugimoto, Hayato Tateda, Hidekazu Takahashi, and Masatoshi Okutomi. Obstacle detection using millimeter-wave radar and its visualization on image sequence. In *IEEE International Conference on Pattern Recognition (ICPR)*, pages 342–345, Cambridge, UK, 2004.

[Tel01] Joel Tellinghuisen. Statistical error propagation. *The Journal of Physical Chemistry A*, 105(15):3917–3921, 2001.

[TK13] Levente Tamas and Zoltan Kato. Targetless calibration of a lidar - perspective camera pair. In *IEEE International Conference on Computer Vision (ICCV) Workshops*, pages 668–675, Sydney, Australia, 2013.

[TX02] Hirohisa Teramoto and Gang Xu. Camera calibration by a single image of balls: From conics to the absolute conic. In *Asian Conference on Computer Vision (ACCV)*, pages 499–506, Melbourne, Australia, 2002.

[TZ00] Phillip H. S. Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138 – 156, 2000.

[VBN12] Francisco Vasconcelos, Joao P. Barreto, and Urbano Nunes. A minimal solution for the extrinsic calibration of a camera and a laser-rangefinder. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2097–2107, 2012.

[Vin11] John Vince. *Quaternions for computer graphics*. Springer, London, UK, 1. edition, 2011.

[VSMH14] Martin Velas, Michal Spanel, Zdenek Materna, and Adam Herout. Calibration of rgb camera with velodyne lidar. In *Conference on Computer Graphics, Visualization and Computer Vision*, pages 135–144, Pilsen, Czech Republic, 2014.

[VW97] Paul Viola and William M. Wells. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.

[Was13] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, New York, USA, 2013.

[WLH$^+$04] N. Williams, Kok-Lim Low, Chad Hantak, Marc Pollefeys, and Anselmo Lastra. Automatic image alignment for 3d environment modeling. In *Brazilian Symposium on Computer Graphics and Image Processing*, pages 388– 395, Curitiba, Brazil, 2004.

[WZXM11] Tao Wang, Nanning Zheng, Jingmin Xin, and Zheng Ma. Integrating millimeter wave radar with a monocular vision sensor for on-road obstacle detection applications. *Sensors*, 11(9):8992–9008, 2011.

[YPIK90] H. K. Yuen, John Princen, John Illingworth, and Josef Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77, 1990.

[ZD12] Lipu Zhou and Zhidong Deng. Extrinsic calibration of a camera and a lidar based on decoupling the rotation from the translation. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 642–648, Madrid, Spain, 2012.

[ZGN01] Xiang Zhang, Yakup Genc, and Nassir Navab. Mobile computing and industrial augmented reality for real-time data access. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 583–588, Juan Le Pins, France, 2001.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(11):1330–1334, 2000.

[ZIMP14] Kun Zhao, Uri Iurgel, Mirko Meuter, and Josef Pauli. An automatic online camera calibration system for vehicular applications. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1490–1492, Qingdao, China, 2014.

[ZP04] Qilong Zhang and Robert Pless. Extrinsic calibration of a camera and laser range finder (improves camera calibration). In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2301–2306, Sendai, Japan, 2004.

Publications by the Author

[1] Tilman Kühner and Julius Kümmerle. Extrinsic multi sensor calibration under uncertainties. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3921–3927, Auckland, New Zealand, 2019.

[2] Tilman Kühner and Julius Kümmerle. Large-scale volumetric scene reconstruction using lidar. In *IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020. In press.

[3] Julius Kümmerle, Timo Hinzmann, Anurag S. Vempati, and Roland Siegwart. Real-time detection and tracking of multiple humans from high bird's-eye views in the visual and infrared spectrum. In *Advances in Visual Computing*, pages 545–556, Las Vegas, USA, 2016.

[4] Julius Kümmerle and Tilman Kühner. Fast and precise visual rear axle calibration. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3942–3947, Auckland, New Zealand, 2019.

[5] Julius Kümmerle and Tilman Kühner. Unified intrinsic and extrinsic camera and lidar calibration under uncertainties. In *IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, 2020. In press.

[6] Julius Kümmerle, Tilman Kühner, and Martin Lauer. Automatic calibration of multiple cameras and depth sensors with a spherical target. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, Madrid, Spain, 2018.

[7] Julius Kümmerle, Marc Sons, Fabian Poggenhans, Tilman Kühner, Martin Lauer, and Christoph Stiller. Accurate and efficient self-localization on roads using basic geometric primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5965–5971, Montreal, Canada, 2019.

Supervised Theses

Ole Berger. 3D object detection by fusing lidar- and camera data. Master's Thesis. Karlsruhe Institute of Technology, 2018-06.

Sebastian Sinn. Automatischer Reglerentwurf für Flugmodelle. Bachelor's Thesis. Karlsruhe Institute of Technology, 2019-10.