Camera Calibration with Non-Central Local Camera Models

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

Dipl.-Ing. Johannes Beck

Tag der mündlichen Prüfung:	10.12.2020

Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr.-Ing. Stefan Hinz

Danksagung

Diese Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mess- und Regelungstechnik (MRT) am KIT entstanden.

Zuallererst möchte ich Prof. Dr.-Ing. Stiller danken, dass ich diese Arbeit am MRT anfertigen konnte. Ohne die guten Rahmenbedingungen am Institut wäre diese Arbeit so nicht möglich gewesen. Ich bedanke mich auch bei Prof. Dr.-Ing. Hinz für die Übernahme des Korreferates.

Mein Dank gilt auch allen Mitarbeitern des Instituts. Die gute Arbeitsatmosphäre, die gemeinsamen Ausflüge, der Besuch von Konferenzen, die wissenschaftlichen Seminare und die dadurch entstandenen Diskussionen, Ideen und Anregungen trugen einen wichtigen Teil zu dieser Arbeit bei.

Ich möchte mich an dieser Stelle auch bei den Probelesern meiner Arbeit, Christian Kinzig, Jan-Hendrik Pauls, Tobias Strauß, Sascha Wirges und Florian Wirth bedanken. Für das Korrekturlesen der mathematischen Kapitel danke ich Johannes Fischer und Sven Richter und für das Überprüfen der Referenzen Maximilian Naumann. Für die sprachliche Korrektur möchte ich mich bei Sonja Schemenauer (komplette Arbeit) und Bastian Schroth (Einleitung und Schluss) bedanken.

Ich bedanke mich auch bei der elektrischen Werkstatt des MRTs, Marcus Hoffner und Thomas Lobe, sowie der mechanischen Werkstatt, Günter Barth, Goran Cicak und Dieter Feix, die mich während meiner Doktorandenzeit bei kleineren und größeren Dingen immer unterstützt haben.

Dem Sekretariat, Sieglinde Klimesch, Erna Nagler und Alexandra Stotz, danke ich für das Abnehmen vieler Verwaltungsaufgaben und Werner Paal für die Unterstützung in allen IT-Angelegenheiten.

Mein ganz besonderer Dank gilt zuletzt meiner Frau Anke und unseren drei Kindern David, Naemi und Helen, denen ich diese Arbeit widme.

Zusammenfassung

Kamerakalibrierung ist eine wichtige Grundvoraussetzung für viele Computer-Vision-Algorithmen wie Stereo-Vision und visuelle Odometrie. Das Ziel der Kamerakalibrierung besteht darin, sowohl die örtliche Lage der Kameras als auch deren Abbildungsmodell zu bestimmen. Das Abbildungsmodell einer Kamera beschreibt den Zusammenhang zwischen der 3D-Welt und der Bildebene.

Aktuell werden häufig einfache globale Kamera-Modelle in einem Kalibrierprozess geschätzt, welcher mit vergleichsweise geringem Aufwand und einer großen Fehlertoleranz durchgeführt werden kann. Um das resultierende Kameramodell zu bewerten, wird in der Regel der Rückprojektionsfehler als Maß herangezogen. Jedoch können auch einfache Kameramodelle, die das Abbildungsverhalten von optischen Systemen nicht präzise beschreiben können, niedrige Rückprojektionsfehler erzielen. Dies führt dazu, dass immer wieder schlecht kalibrierte Kameramodelle nicht als solche identifiziert werden.

Um dem entgegen zu wirken, wird in dieser Arbeit ein neues kontinuierliches nicht-zentrales Kameramodell basierend auf B-Splines vorgeschlagen. Dieses Abbildungsmodell ermöglicht es, verschiedene Objektive und nicht-zentrale Verschiebungen, die zum Beispiel durch eine Platzierung der Kamera hinter einer Windschutzscheibe entstehen, akkurat abzubilden. Trotz der allgemeinen Modellierung kann dieses Kameramodell durch einen einfach zu verwendenden Schachbrett-Kalibrierprozess geschätzt werden.

Um Kalibrierergebnisse zu bewerten, wird anstelle des mittleren Rückprojektionsfehlers ein Kalibrier-Benchmark vorgeschlagen. Die Grundwahrheit des Kameramodells wird durch ein diskretes Sichtstrahlen-basiertes Modell beschrieben. Um dieses Modell zu schätzen, wird ein Kalibrierprozess vorgestellt, welches ein aktives Display als Ziel verwendet. Dabei wird eine lokale Parametrisierung für die Sichtstrahlen vorgestellt und ein Weg aufgezeigt, die Oberfläche des Displays zusammen mit den intrinsischen Kameraparametern zu schätzen. Durch die Schätzung der Oberfläche wird der mittlere Punkt-zu-Linien-Abstand um einen Faktor von mehr als 20 reduziert. Erst dadurch kann das so geschätzte Kameramodell als Grundwahrheit dienen.

Das vorgeschlagene Kameramodell und die dazugehörigen Kalibrierprozesse werden durch eine ausführliche Auswertung in Simulation und in der echten Welt mithilfe des neuen Kalibrier-Benchmarks bewertet. Es wird gezeigt, dass selbst in dem vereinfachten Fall einer ebenen Glasscheibe, die vor der Kamera platziert ist, das vorgeschlagene Modell sowohl einem zentralen als auch einem nicht-zentralen globalen Kameramodell überlegen ist. Am Ende wird die Praxistauglichkeit des vorgeschlagenen Modells bewiesen, indem ein automatisches Fahrzeug kalibriert wird, das mit sechs Kameras ausgestattet ist, welche in unterschiedliche Richtungen zeigen. Der mittlere Rückprojektionsfehler verringert sich durch das neue Modell bei allen Kameras um den Faktor zwei bis drei.

Der Kalibrier-Benchmark ermöglicht es in Zukunft, die Ergebnisse verschiedener Kalibrierverfahren miteinander zu vergleichen und die Genauigkeit des geschätzten Kameramodells mithilfe der Grundwahrheit akkurat zu bestimmen. Die Verringerung des Kalibrierfehlers durch das neue vorgeschlagene Kameramodell hilft die Genauigkeit weiterführender Algorithmen wie Stereo-Vision, visuelle Odometrie oder 3D-Rekonstruktion zu erhöhen.

Abstract

Camera calibration is a crucial prerequisite for most computer vision tasks such as stereo vision and visual odometry. The goal of camera calibration is to find the transformation between multiple cameras as well as the parameters of a camera model which describes how the 3D world is projected onto the 2D image sensor.

Today's choice of calibration processes often focuses on ease of use, thus, using simple camera models. Generally after calibration, the performance of these models is assessed using the RMS reprojection error. However, even simple camera models which cannot accurately describe the optical system may achieve small RMS reprojection errors, hiding the insufficiency of the estimated model.

To overcome these issues, a novel continuous non-central camera model is proposed. This camera model is able to cope with various lenses and non-central distortions induced by additional optical components, such as windshields, placed in front of the lens. The camera model is based on uniform B-splines which makes it efficient to compute and very flexible. While being more powerful than simpler camera models, its parameters can still be estimated using a fully automated, easy-to-use and robust checkerboard calibration process.

Instead of using the RMS reprojection error metric for performance assessment, a calibration benchmark is presented. The ground truth camera model, based on discrete viewing rays, is generated by an active display calibration process. We propose a local parametrization of these viewing rays and a way to jointly estimate the surface of the display and the camera model parameters. Estimating the display surface reduces the mean ray-point distance by a factor of more than 20, enabling the generation of true ground truth camera models.

The proposed camera model and its calibration processes are extensively evaluated in simulation and, leveraging the calibration benchmark, the proposed camera model is evaluated in the real world. The evaluation shows that the novel camera model outperforms both a global central and a non-central camera model even in the simple case of a glass pane placed in front of the lens. Finally, the practicability of this model is shown by calibrating an experimental vehicle with a surround-view setup consisting of six cameras, reducing the RMS reprojection error by a factor of two to three for all cameras.

The calibration benchmark enables researchers to compare calibration algorithms and to assess the quality of camera models accurately using a verified ground truth. The lowered calibration error of the proposed camera model helps downstream algorithms such as stereo vision, visual odometry and 3D-reconstruction to improve the overall system performance.

Contents

Notation and Symbols

Acronym

BCM	generic B-spline distortion camera model
CMDI	camera model difference
DCM	discrete camera model
DOF	degrees of freedom
EVD	eigenvalue decomposition
FOV	field of view
GCM	generic global camera model
KIT	Karlsruhe Institute of Technology
LLS	linear least squares
LS	least squares
MAE	mean absolute error
NLS	non-linear least squares
PCA	principal component analysis
REH	reprojection error histogram
RMS	root-mean-square
RMSE	root-mean-square error
ROI	region of interest
SVD	singular-value decomposition

General Notation

Scalars	Regular (greek) lower case	a, b, c, α, β
Vectors	Bold lower case	$\mathbf{a}, \mathbf{b}, \mathbf{c}$
Matrices	Bold upper case	$\mathbf{A}, \mathbf{B}, \mathbf{C}$
Numbers	Blackboard upper case	$\mathbb{N}, \mathbb{R}, \mathbb{H}$

Indexing

x_i	i^{th} element of vector $\mathbf{x}$
A_{ij}	$(i, j)^{\text{th}}$ element of $\mathbf{A}$
$\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2)$	Matrix $\mathbf{A}$ composed of columns $\mathbf{a}_1$ and $\mathbf{a}_2$

Numbers and Sets

$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathbb{H}$	Quaternion numbers
$\mathbb{P}^n$	n-d projective space
$\mathbb{S}^n$	n-sphere space

Mathematical Operators

$\mathbf{x} \cdot \mathbf{y}$	Dot product of $\mathbf{x}$ and $\mathbf{y}$
$\lvert\cdot\rvert$	Absolute value
$\lVert\cdot\rVert$	Vector 2-norm
$\lfloor\cdot\rfloor$	Floor function
$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$	Gradient (Jacobian matrix) of $\mathbf{f}$ with respect to $\mathbf{x}$
$\mathbf{f}^{(n)}(t)$	n^{th} derivative of $\mathbf{f}$
$(\cdot)^+$	Moore-Penrose inverse (pseudo inverse)

Coordinate Systems

$\mathbf{u} = (u, v)^\top$	2D point
$\mathbf{p} = (p_x, p_y, p_z)^\top$	3D point
$\{\mathrm{A}\}$	Coordinate system A
$\mathbf{p}^{\mathrm{A}}$	Point $\mathbf{p}$ in coordinate system A
$\mathbf{T}^{\mathrm{BA}}$	Transformation matrix $\mathbf{T}^{\mathrm{BA}}$ to transform points from $\{\mathrm{A}\}$ to $\{\mathrm{B}\}$

Camera Model

$\mathbf{x}_0$	Base point of a line
$\mathbf{d}$	Direction vector of a line
$\mathcal{P}$	Camera model
$\mathcal{P}_{\mathrm{F}}$	Forward camera model
$\mathcal{P}_{\mathrm{B}}$	Backward camera model
$\mathcal{P}_{\mathrm{B,X}}$	Base point part of backward camera model
$\mathcal{P}_{\mathrm{B,D}}$	Directional part of backward camera model

1 Introduction

Cameras are present everywhere in today's life and are commonly used to take pictures, stream live videos and enable video chat. Even low-cost cameras provide good image quality and new cameras with higher resolutions and lower prices reach the market every year.

In order to increase throughput, safety, reliability and comfort for humans, cameras used as measurement tools have become increasingly interesting to automate tedious and monotonic work. Cameras can already be found in various work settings such as manufacturing lines for product inspection, mills to sort out bad from good seed, in drones to inspect construction sites, augmented reality to identify buildings and plants or humanoid robots and self-driving cars.

Speaking of self-driving cars, due to recent breakthroughs in machine learning with neural networks, the camera is nowadays a well-established sensor among range sensors, radars and inertial measurement units.

For a lot of these tasks, the knowledge of solely the camera image, meaning the intensity value of each camera pixel, is not sufficient. Additionally, the position and orientation of the camera in relation to other sensors and the information of how the real world is projected onto the image sensor is required. The goal of camera calibration is to determine these quantities. The transformation between cameras is usually described by an affine transformation and the projection is described by a mathematical model which is called the camera model. The parameters that are used to describe the transformation are called extrinsic and those used to describe the camera model are called intrinsic camera parameters.

A camera model is fully defined by using either a forward or a backward model. Forward models project a 3D point in space into the 2D image space. Backward models map a 2D image point to a line in 3D which is called a viewing ray.

1.1 Problem Statement

Ideally, the camera model would be mathematically described by a set of discrete viewing rays. The camera calibration process would have a simple setup and enable a fully automated estimation of the intrinsic and extrinsic camera parameters in reasonable time. Additionally, the determined parameters would accurately describe the real projection model of the camera. In the context of self-driving cars, recording a calibration sequence simply by driving around in regular traffic and then feeding the recorded sequence into a calibration algorithm which determines the calibration parameters accurately without manual intervention would be the ‘holy grail’.

Today most calibration processes are designed with the emphasis that they be favorable in both setting up and performing the calibration and still deliver acceptable accuracy. This is achieved by using special 2D targets printed by widely available printers and hand-held recorded calibration sequences followed by a fully automated extraction and estimation process where only a few initial parameters need to be set by the user. Often simple camera models with a one-digit number of parameters are used and it is assumed that all rays hitting the image sensor pass through a single point.

The downside of this calibration process is that it is almost impossible to be certain about the quality of the calibration results. To discern the quality of a camera calibration, various calibration steps need to be inspected manually for which expert knowledge is required and still even for experts it can be hard to assess the quality of the estimated calibration. In reality, a camera calibration is often performed by non-experts, only simple measures like the reprojection error are used and a calibration is accepted as suitable if the root-mean-square (RMS) reprojection error is below a certain threshold.

As a consequence, in downstream computer vision algorithms a suboptimal camera model is trusted without directly taking any calibration error into account. This leads to algorithms which work around the calibration error leading to less-than-ideal performance regarding accuracy or execution time.

In this thesis, we take a step towards the ‘holy grail’ of camera calibration by introducing a more flexible camera model which can be estimated by an easy-to-use calibration setup and by providing a calibration benchmark to assess calibration performance.

1.2 Contributions

The first major contribution of this thesis is a novel non-central local camera model that uses a minimal-parametric continuous representation of the backward camera model based on splines with local support:

- This camera model can handle various types of distortions of different lens types even if the camera is placed behind a windshield - a common situation in autonomous driving.
- This camera model can be integrated into an easy-to-use and fully automated calibration process using checkerboard targets.
- Due to the explicit backward camera model and implicit forward camera model formulation, we show how the intrinsic camera parameters can be estimated by using an optimal residual (reprojection error) in a non-linear least squares (NLS) problem formulation by deriving the Jacobian matrix of implicit forward camera models.
- We propose a computationally efficient camera model prior based on the smoothness of splines which increases the stability of the estimation algorithm.
- The superior performance of the camera model is demonstrated by an extensive evaluation in simulation, in a calibration benchmark and in a real-world scenario consisting of multiple cameras in the context of autonomous driving.

The second major contribution is the proposal of a calibration benchmark:

- We show how to obtain a ground truth camera model. We use independent viewing rays for each pixel which are estimated by using an active display calibration process. In this process, the non-planarity of the display surface is taken into account, which increases the accuracy to a level which makes it suitable for a ground truth camera model.
- We propose a local line parametrization needed to optimize viewing rays in a NLS problem.
- We propose a distance measure between two camera models to compare the ground truth model with the estimated model.

1.3 Outline

In **Chapter 2** we present an overview of related work in the field of camera calibration.

This is followed by the theoretical basics in **Chapter 3** which are the fundamentals for the following chapters to draw on. This chapter starts with the description of splines followed by various sections belonging to the topic of estimation. We describe and compare different spline types, show how to formulate and solve estimation problems and present different parametrizations of optimization variables.

In **Chapter 4** different central and non-central camera models are introduced. We start with global camera models, where a few parameters are sufficient to describe the full model, followed by a discrete camera model, where the camera model is described by a set of discrete viewing rays, and finally the novel B-spline distortion camera model. We discuss how to derive the Jacobian matrix of an implicit forward camera model function and how a windshield in front of a camera influences the camera model.

To actually determine the parameters of a camera model, two different calibration processes are presented in **Chapter 5**. The first one focuses on accuracy using a display and a three-axis linear positioning system in which a discrete camera model can be calibrated. This process provides a highly accurate camera model when the display is modeled as a non-flat surface. The second calibration process lays its focus on simplicity and flexibility. Here checkerboards are used as targets. For recording the input sequence the camera can be moved around freely without measuring the poses. This calibration process is capable of estimating global camera models as well as the proposed B-spline distortion model.

In **Chapter 6** we discuss methods to assess the performance of camera calibrations. We present the reprojection error histogram (REH) and the camera model difference (CMDI) which is used to quantify the difference between two camera models.

The proposed novel camera model and its calibration processes are extensively evaluated in **Chapter 7**. For both calibration processes introduced in Chapter 5, the evaluation is first done in simulation. Afterwards, we present a calibration benchmark using the *triple camera setup* for which the ground truth camera model is generated by the active display camera calibration. This is used to benchmark a global camera model as well as the proposed local B-spline distortion camera model using the checkerboard calibration process. At the end we demonstrate the gain in performance when using the B-spline distortion model to calibrate our autonomous vehicle with a surround-view camera setup.

We finish the thesis with a conclusion and an outlook in **Chapter 8**.

2 Related Work

In this chapter, we present other work related to camera calibration. We limit the scope to target-based calibration processes in which a complete camera model is estimated and not only, e.g., distortion parameters. We start with a brief history followed by the state of the art.

2.1 History

Camera calibration is a fundamental computer vision problem with a long history. There are mainly two different modeling approaches: camera calibration using physical models and camera calibration using mathematical models. With physical models we mean a modeling approach which tries to describe the underlaying projection model of the lens, and with mathematical models we mean a modeling approach which uses a mathematical function that has no relationship to, e.g., any lens parameters.

Calibration using Physical Models

Back in 1972, Sobel [Sob70] described how a pinhole camera model without lens distortions may be calibrated. He used four known 3D triangles on a plane and showed that the intrinsic camera parameters can be estimated by solving linear equations.

In 1987, Tsai [Tsa87] showed how to calibrate a pinhole camera model with distortions but with a known principal point by using known marker positions on a plane. The measurements for calibration were created by using the corners of squares positioned on a flat surface. For estimation, he presented a two-stage approach: First, the 3D orientation and the extrinsic xy coordinates were estimated. Second, the intrinsic camera parameters and the z position were determined. This approach was then extended to use multiple views of the calibration target whose positions were measured by a Klinger vertical micrometer stage. This was further extended by Lenz et. al. [LT88] to estimate the principal point.

In 1992, Weng et. al. [WCH92] used a different improved two-stage algorithm. In the first step, the ideal camera model without distortion parameters and the extrinsic parameters were estimated in closed form. In the second step, the full intrinsic camera model with distortions and the extrinsic pose were determined jointly by minimizing the reprojection error using a non-linear least squares solver.

Up to this point, the target geometry and the movements of the target or camera were measured by an external device such as robot arms. In 1999 and 2000, Sturm et. al. [SM99] and Zhang [Zha99, Zha00] independently presented a way of estimating the target poses. Sturm used a simplified camera model and homographies to form a linear equation system and then determined the intrinsic camera parameters. In contrast, Zhang showed how to estimate the intrinsic and extrinsic parameters, including distortions, and the target poses by a non-linear least squares solver minimizing the reprojection error. As targets he used the corners of black squares on a surface.

Due to its simplicity, the same concept is still used and implemented with all kinds of modifications such as using different targets up to this day, e.g., in the popular OpenCV library [Bra00].

Calibration using Mathematical Models

In 1981, Martins et. al. [MBK81] presented one of the first non-central local camera models. To generate measurements, they used a single circular white target which was moved by a robot arm in two different calibration planes. For each position, the image location was estimated. The camera model was described by a set of lines defined by the hit points of the two calibration planes. Since sparse measurements were used, different interpolation methods, namely linear, quadratic and linear spline interpolation, were employed for each plane. This method is also called the ‘two-planes method’. In 1988, Gremban et. al. [GTK88] used the two-plane model to determine a complete camera model and showed how to estimate pinhole camera model parameters based on a set of lines.

In 1992, Champleboux et. al. [CLSC92] extended the work of Martins by formalizing the mathematical model by a function $\mathrm{f}\colon \mathbb{R}^2 \rightarrow \mathbb{R}^4, (u, v) \mapsto (x_1, y_1, x_2, y_2)$ where (u, v) are image coordinates and (x_1, y_1), (x_2, y_2) are positions on two distinct planes. They modeled this function as a bicubic B-spline and used multiple planes for calibration.

In 2001, Grossberg et. al. [GN01] formalized a camera model in which a discrete incoming viewing ray is mapped to 'photo-sensitive elements on the image detector' [GN01]. These photo-sensitive elements which they called 'raxels' included geometric, radiometric and optical properties. A display was used for calibration. Different binary-coded images were displayed on an LCD and images were recorded by a camera at two different locations. The movement between these two locations was measured manually. In later works, this camera model was estimated either to cover the whole field of view (FOV), or without knowing the motions between different views, or without the use of active targets (see, e.g., [SR04, GN05, RSL05, RW12, SLPS20]).

2.2 State of the Art

In this section, we summarize the state of the art of camera calibration, split up into four main subsections, namely the camera models, the inversion of camera models, target surface estimation and calibration performance assessment.

2.2.1 Camera Models

A camera model is fully defined by using either a forward or a backward model. Forward models project a 3D point in space onto the 2D image space. Backward models map a 2D image point to a line in 3D which is called a viewing ray (see Chapter 4).

In literature, a lot of different camera models have been proposed. We refer to [SRT+11, Han11, PBSG12, ZZH13, Lub15, LRKB19] for an extensive overview. We split our review into two sections regarding how the viewing ray directions are described and how the viewing ray base point is modeled.

Viewing Ray Direction Modeling

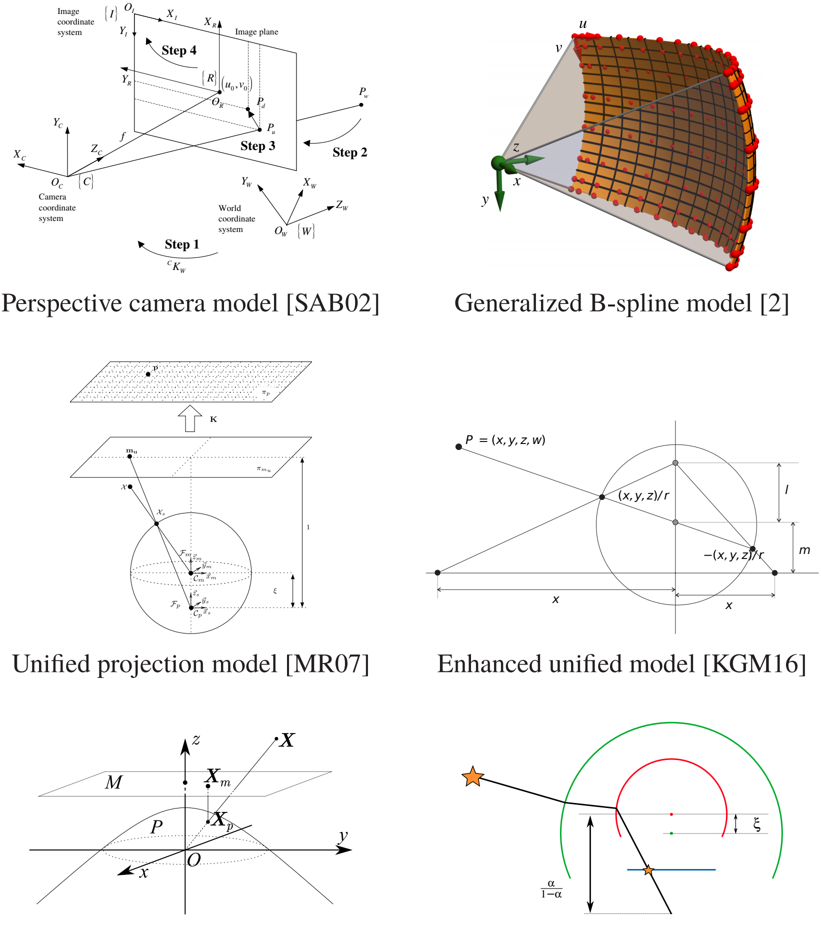
The modeling of the viewing ray direction can be physically motivated or simply a mathematical model. Physically motivated camera models such as global or catadioptric camera models tend to have a low number of parameters. In general, they are easy to use, and robust calibration methods exist. In contrast, mathematical models tend to have more parameters but they are not specific to a single optical system. We will give an overview of such camera models (some of these models are visualized in Fig. 2.1):

Perspective Global Camera Models The fundamental model for perspective lenses is the pinhole camera model. As it is not possible to build a perfect perspective lens, distortions parameters are added to the pinhole camera model. A lot of different distortion models have been proposed. A popular choice consists of even polynomial radial distortions and decentering distortions which are a type of tangential distortions, e.g., in [Bro66, Tsa87, Zha99] in forward projection and, e.g., in [WCH92] in backward projection. Instead of using even polynomial radial distortions, rational polynomials are proposed in [Fit01, BBV01] in a special case in backward direction and in [MCM04] in forward direction. For tangential distortions, the decentering and tilt may be described directly [WSZL08].

Wide-angle Global Camera Models To generalize the pinhole camera model to a wider FOV and to describe the distortions induced by lenses with larger FOVs, a function which takes the angle between the optical axis and the viewing ray as input and outputs the distance to the principal point is introduced. Four different functions can be identified, namely the stereographic, equidistant or equiangular, orthogonal, and equisolid projections [Bec25, KB06].

As wide-angle lenses tend to have stronger distortions than perspective lenses, different radial distortions models have been presented. In [BL93, BL95], the polynomial fish-eye transform was proposed[1]. The radial distortions are modeled by a polynomial series including odd and even terms. In [ZWY11], an elliptical function model using lifted coordinates was proposed. Polynomial tangential distortions were proposed in [SA96]. The rational radial polynomial distortions function of [Fit01, BBV01] was also used for wide-angle lenses.

[1] Even though the camera model is called ‘fish-eye transform’, it cannot handle lenses with a FOV $\geq 180°$.



Perspective camera model [SAB02]

Generalized B-spline model [2]

Unified projection model [MR07]

Enhanced unified model [KGM16]

Sphere model [GD00]

Double sphere model [UDC18]

Figure 2.1: Visualization of different camera models proposed in literature.

This list contains only a few examples. More distortion models can be found, e.g., in [HS96, KB00, DF01, BP02].

Catadioptric Camera Models Camera models for catadioptric lenses differ mainly in which types of mirrors can be modeled, in the projection model, and in how the intrinsic and extrinsic parameters are estimated. The unified camera model proposed in [MR07] showed a camera model which can handle different mirror shapes. The world point is projected onto a unit sphere, transformed into the camera coordinate system, projected onto a normalized plane and then projected onto the image plane using a generalized camera projection matrix. This model was further extended in [KGM16]. Other models like the sphere camera model [GD00] and the double sphere model [UDC18] were also proposed. A survey can be found in [PBSG12, ZZH13].

Unifying Global Camera Models There are attempts to describe generic camera models which allow the use of a single mathematical framework for different lens types. Central catadioptric and fisheye lenses were modeled within a unified camera model in [YH04, SMS06], a rational function lens distortion model for general cameras was described in [CF05], a generalized camera model including fish-eye lenses was proposed in [Gen06] and a general linear camera which models perspective, orthographic, and many multi-perspective cameras was presented in [YM04].

Discrete and Local Camera Models Discrete camera models are modeled with a single ray per pixel. To estimate such a model, at least two 3D measurements per pixel are needed. This is achieved by using a display or sparse measurement like checkerboards corners. In the case of active display camera calibration, either the poses of the display need to be measured externally [GN01, GN05, HSK16] or the poses are estimated during the calibration process [BART13]. In case of sparse measurements, there are two approaches: either the sparse measurements or the discrete rays are interpolated. In [SR04, RSL05], the measurements were interpolated using homographies and in [MBK81], using linear, quadratic and spline functions. Spline functions were also used in [CLSC92]. For interpolating discrete viewing rays, rational basis splines in Plücker coordinates [MAQ11, MA13] or B-splines [VT05, SLPS20, 2] were proposed. In [RW12], the measurements as well as the discrete rays were interpolated using B-splines. This camera model can be further constrained to central camera models to reduce complexity and increase robustness [DMW10].

Viewing Ray Base Point Modeling

A camera model is called non-central if there is no single point through which all viewing rays pass. This means that the base points of all viewing rays are not the same. Sometimes the term ‘non-single viewpoint’ is used instead of ‘non-central’.

Non-central camera models are far less common than central camera models. We split the discussion into global, catadioptric, discrete, and local camera models.

Global Non-central Camera Models In the context of fisheye lenses, Gennery [Gen06] proposed to use a radially symmetric shift of the base point along the optical axis. He analysed different fisheye lens designs and modeled the relationship between the viewing ray angle θ and the displacement s along the optical axis as $s = \frac{\theta}{\sin(\theta)} - 1$. This formula is extended by a polynomial series of even powers to account for further deviations.

Catadioptric Non-central Camera Models Some catadioptric camera models are by design non-central. E.g., the double sphere model [UDC18] is by design non-central. The effect of the non-central property can be described by a caustic. ‘With respect to imaging devices, caustics represent their loci of viewpoints’ [SGN06]. In [SGN06], the shapes of the caustic for different catadioptric lenses were analyzed and parametric representations were derived.

Discrete Camera Models In the general discrete camera model proposed in [GN01, GN05], the base point is modeled as a point in 3D. This model is capable of describing all sorts of different optical systems.

Local Camera Models Local camera models interpolate the base point between a few viewing rays. In case of the general linear camera model [YM04], these viewing rays are interpolated using affine transformations. Another methods of interpolation use rational basis splines [MAQ11, MA13] or B-splines [RW12] in Plücker coordinates. In [SLPS20], the 3D base points are directly interpolated using uniform B-splines.

Comparison with the Proposed B-spline Distortion Camera Model

Unfortunately, every single one of these models is either specifically built to fit a special central or non-central lens or else it is not minimal-parametric, meaning that the viewing rays are described with more than four parameters. In this thesis, we propose a continuous minimal-parametric uniform B-spline camera model which is not lens-specific.

2.2.2 Inversion of Camera Models

Usually either the forward or the backward camera model can be expressed in closed form. For some camera models, there are methods to calculate the inverse model efficiently. In [MCM06, MCM04, Hei00, MW04], different ways of inverting a pinhole camera model with distortions are proposed. For catadioptric lenses, special non-central forward projection models are proposed in [ATR10, ATR11] and [GN09, Gon10]. To project a point from 3D onto the image, the roots of a polynomial with degree between four and twelve need to be found.

For generic non-central camera models, modeling the backward projection is usually the only possibility. Therefore, the forward camera model is only defined implicitly. If the Jacobian matrix of the camera model is needed during a calibration process numeric differentiation is used (see, e.g., [SLPS20]). This leads to sometimes unstable optimization algorithms, and numeric differentiation is computationally expensive. To the best of our knowledge, an analytic expression of the Jacobian matrix of an implicit forward camera model has so far not been proposed.

2.2.3 Target Surface Estimation

For active display calibration, the assumption of a flat display surface does not hold true and therefore leads to a less accurate discrete camera model. In literature, this effect is neglected and often not even recognized. In [BCG+17], some artifacts in the accumulated point-line distance image were noticed and it was guessed that they might originate in the non-planarity of the display. But in the end, the artifacts were not further investigated and the surface of the display was neither modeled nor estimated.

For checkerboard calibration targets, there is some research on how to deal with imperfect targets. In, e.g., [SH11, HZA13, Str15], for each 3D checkerboard corner a 3D offset for each corner was jointly estimated with the extrinsic and intrinsic camera parameters. In [BNPR19, BNPR20], a stereo camera setup was calibrated by alternating between the estimation of the intrinsic and the extrinsic camera parameters using constant corner positions on the one hand and the estimation of the 3D corner positions without optimizing the camera positions on the other hand.

In the context of structured-light 3D surface imaging, the calibration is assumed to be known beforehand and only surfaces are reconstructed [Gen11].

To the best of our knowledge, a continuous display surface jointly estimated with a discrete camera model has not been proposed before.

2.2.4 Calibration Performance Assessment

The performance of a camera calibration is frequently assessed by the minimum of the error function. The error metric used to build the error function is either in image space, then called reprojection error, or in world space, then called ray-point distance. In addition, two other measures are used, namely the parameter uncertainty and the performance difference of applications when using different camera models.

Reprojection error The reprojection error is the difference between the measured target position in the image and the 3D target position projected onto the image using the forward camera model. As there are multiple measurements, different statistical approaches are used. The RMS reprojection error calculated on the calibration set and sometimes also on a test set is the most used error metric, e.g., in [KB06, SMS06, RW12, BART13, KGM16, ZSJ+17, UDC18, SLPS20]. To get more insight on the error distributions, other statistical measures are used. E.g., in [KGM16], the standard deviation of the reprojection error is calculated, a box-plot is used in [Sch14] and a error histogram is used in [Str15, BNPR20, 2]. In [SLPS20], the distribution of the reprojection errors at different image locations is inspected and compared with a Gaussian distribution using the Kullback-Leibler divergence.

Ray-point distance The ray-point distance is the shortest distance between the viewing ray, calculated using the backward camera model at the 2D image position of the target position, and the 3D target position. The mean ray-point distance is used for performance assessment in, e.g., [RW12, MA13, Ros16].

Parameter uncertainty Another way of performance assessment is to determine the uncertainty of the estimated parameters. In [Zha00], the uncertainty of the intrinsic camera parameters was determined by comparing the intrinsic camera model parameters of three calibrations created from three different input sequences. In [Ros16], the uncertainties of the estimated ray parameters were presented.

Performance assessment using applications The performance of a camera calibration may also be assessed by inspecting the performance difference in applications in which the camera model plays an important role. A 3D reconstruction of a maize seedling is shown in [ZSJ+17]. A 3D reconstruction of a Bas-relief and an injection mould for a shoe sole is shown in [BNPR20]. In [SH11], the consistency of the difference between two 3D points is used. In [SMS06], a 3D reconstruction experiment with a trihedron is used and in [Sch14] a localization experiment is performed. In [WCH92], the normalized stereo calibration error is proposed which tries to build an error metric independent of stereo camera setup design parameters, such as FOV or base line distance.

To the best of our knowledge, a calibration benchmark in which a generic camera model is used as ground truth that is then compared with different camera models has not been proposed before.

3 Theoretical Basics

This chapter introduces the theoretical basics used throughout the thesis. The main tools used are splines and parameter estimation, which are covered in Sections 3.1 and 3.2. The spline section gives an overview of different splines and a comparison of the different spline bases. The estimation chapter briefly discusses linear least squares (LLS) and non-linear least squares (NLS) problems and then focuses on how parameters like rotations and lines in 3D can be used in such optimization problems. Also, line fitting problems are shown where a closed solution exists. Section 3.2 ends by explaining how splines and spline priors are integrated into a NLS problem.

Some sections only briefly introduce the topic and then refer to other literature with an in-depth description. The goal is to provide an overview to which we reference in subsequent chapters.

3.1 Splines

We start with an example to motivate splines before giving a more precise mathematical definition. Suppose given N_c distinct 2D points $\mathbf{z}_i = (z_{i,x}, z_{i,y})^\top, i = 1, \ldots, N_c$, we search a function $\mathbf{s}(t)$ which passes through or is close to these points. A first idea might be a parametric polynomial function of degree $N_c - 1$

$$\mathbf{s}(t) = \sum_{i=1}^{N_c} \mathbf{c}_i t^{i-1}, \tag{3.1}$$

where $\mathbf{c}_i = (c_{i,x}, c_{i,y})^\top$ are the polynomial coefficients. To determine the polynomial coefficients we state that the function should pass through the given points. This leads to the constraints

$$\mathbf{s}(i) = \mathbf{z}_i \quad \forall i. \tag{3.2}$$

This is a linear equation system with $2N_c$ equations and $2N_c$ unknowns. The resulting function is of class C^∞ as all derivatives exist and are continuous. The drawback is that for a high number of points the global polynomial function can become unstable (also called the Runge's phenomenon).

So instead of using one global polynomial function we could use a piecewise function

$$\mathbf{s}(t) = \begin{cases} \mathbf{s}_1(t), & t_1 \le t < t_2 \\ \mathbf{s}_2(t), & t_2 \le t < t_3 \\ \vdots & \vdots \\ \mathbf{s}_{N_s}(t), & t_{N_s} \le t \le t_{N_s+1} \end{cases}, \tag{3.3}$$

where N_s is the number of spline segments and $\mathbf{s}_i(t), i = 1, \ldots, N_s$, the spline segment functions.

Using $N_s = N_c - 1$ segments and a straight line for each segment results in

$$\mathbf{s}_i(t) = \mathbf{c}_{i,1}t + \mathbf{c}_{i,2}, \tag{3.4}$$

which satisfies $\mathbf{s}_i(t_i) = \mathbf{z}_i$ and $\mathbf{s}_i(t_{i+1}) = \mathbf{z}_{i+1}$. Choosing $t_i = i{-}1$ leads to

$$\mathbf{s}_i(t) = \mathbf{z}_{i+1}(t - i + 1) + \mathbf{z}_i(i - t). \tag{3.5}$$

This function is a polygonal chain which does not suffer from becoming unstable when using a lot of points. The drawback is that at t_i it is only continuous but not continuously differentiable, so this function is of class C^0.

We have seen two extreme cases, a global polynomial function of C^∞ and a polygonal chain of C^0. Now we will formulate a spline mathematically and show how we can formulate functions between those two extreme cases.

We define splines as polynomial functions $\mathbf{s}\colon \mathbb{R}^{N_i} \to \mathbb{R}^{N_o}$. These functions all have in common that they are defined by the degree of the polynomial d, N_c control points $\mathbf{p}_k \in \mathbb{R}^{N_o}$ and a strictly monotonically increasing vector $\mathbf{t} = (t_1, t_2, \ldots, t_{N_s+1}) \in \mathbb{R}^{N_s+1}, t_i < t_{i+1}, \forall i$. To formulate the spline function

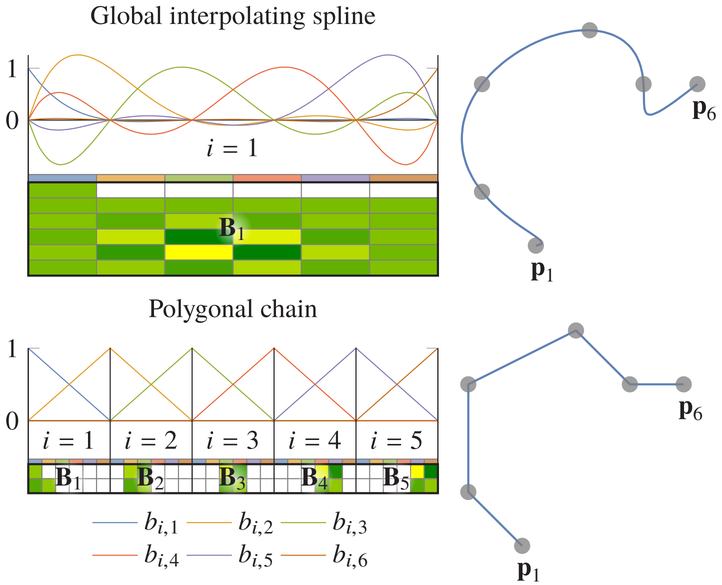


Figure 3.1: Visualization of a global and polygonal chain spline. On the left, the basis matrix $\mathbf{B}_i$ and the basis functions $\mathrm{b}_{i,k}$ for each segment is visualized. On the right, the resulting spline using the control points $\mathbf{p}_i$ is plotted.

for $N_i = 1$ we use the piecewise defined function $\mathbf{s}(t)$ from Eq. (3.3) and define the segment functions $\mathbf{s}_i(t)$ as

$$\mathbf{s}_i(t) = \sum_{k=1}^{N_c} \mathbf{p}_k \, \mathrm{b}_{i,k}(t) = \mathbf{P}\,\mathbf{B}_i^\top \mathbf{a}(t), \tag{3.6}$$

$$\mathbf{a}(t) = \left(1, t, \ldots, t^d\right)^\top, \tag{3.7}$$

$$\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N_c}), \tag{3.8}$$

where $\mathbf{B}_i \in \mathbb{R}^{(d+1)\times N_c}$ is called the spline basis. The characteristics of the spline $\mathbf{s}(t)$ are controlled by the three parameters d, $\mathbf{B}_i$ and $\mathbf{t}$. In order to evaluate the spline, we are most interested in its smoothness, its compactness and its efficiency and numerical stability. We denote the smoothness with the differentiable class C^k, $k \in \mathbb{N}_0 \cup \{\infty\}$, and we call a spline compact if changing a single control point $\mathbf{p}_k$ does not lead to a change of the whole spline. To infer the different properties of the splines we visualize the basis $\mathbf{B}_i$ and the basis function $\mathrm{b}_{i,k}$.

The visualization of the global interpolating spline and the polygonal chain using six control points is shown in Fig. 3.1. For the global interpolating spline there is only one segment and $d = 5$. The basis functions are shown above the spline basis. In this example we have six basis functions and one spline basis $\mathbf{B}_1 \in \mathbb{R}^{6\times6}$. Each column of the spline basis belongs to a control point and each row to a polynomial coefficient. A zero is indicated by a white square, whereas a non-zero entry is colored from dark green (smallest value) to yellow (largest value). As there is a non-zero in every row and column of the matrix, the spline is not compact. For the polygonal chain we have five segments and $d = 1$. Using the same number of control points, the spline basis is $\mathbf{B}_i \in \mathbb{R}^{2\times6}$. For the first segment $i = 1$, the basis matrix $\mathbf{B}_1$ only contains non-zero entries in the first two columns, so the spline value is only dependent on the first two control points $\mathbf{p}_1$ and $\mathbf{p}_2$. For the second segment $i = 2$, the basis matrix $\mathbf{B}_2$ only contains non-zero entries for the second and third columns, so this segment only depends on the control points $\mathbf{p}_2$ and $\mathbf{p}_3$. This is similar for the other segments. Therefore, using such a basis results in a compact spline. From the basis vector plot we see that the resulting spline position is a linear interpolation between the two depending control points. The spline $\mathbf{s}(t)$ for both cases using six arbitrary positioned control points is plotted on the right of Fig. 3.1.

So far the input dimension of the spline was one. To extend it to $N_i = 2$, we for now assume a single segment spline and formulate it as

$$\mathbf{s}_0(x, y) = \sum_{i=1}^{N_{c_1}} \sum_{j=1}^{N_{c_2}} \mathrm{b}_{1,i}(x)\mathrm{b}_{2,j}(y)\mathbf{p}_{ij}. \tag{3.9}$$

The spline function is constructed by two 1D basis functions b_1 and b_2. The control points are now located on a 2D grid. Extending the spline for $N_i > 2$ is possible but not needed for this work as we are only describing surfaces with splines.

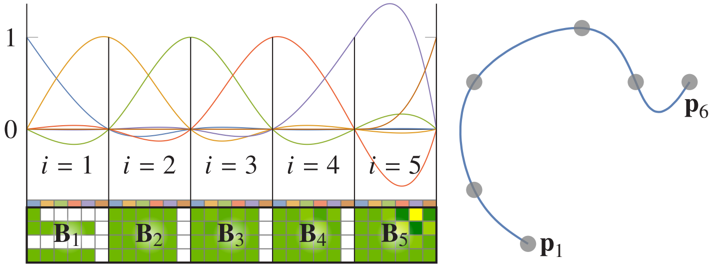


Figure 3.2: Visualization of an interpolating polynomial spline of degree three with six control points. The spline passes through the control points and is not compact, as in general all columns of the basis matrices $\mathbf{B}_i$ are non-zero.

In the following sections, we describe different spline bases and their influences on the resulting spline for $N_i = 1$. We always use intervals of the same size for $\mathbf{t}$:

$$t_{i+1} - t_i = c > 0, \quad i = 1, \ldots, N_c. \tag{3.10}$$

This simplifies the comparison between the splines. In the last Section 3.1.5 the splines are compared to each other. As we are frequently using splines in optimization problems, further information on how spline parameters can be estimated is given in Section 3.2.7.

3.1.1 Interpolating Polynomial Splines

An interpolating polynomial spline

$$\mathbf{s}_i(t) = \sum_{k=0}^{d} \mathbf{c}_{i,k+1} t^k, \tag{3.11}$$

uses $N_s = N_c - 1$ spline segments of degree d and is fully defined by the constraints

$$\mathbf{s}(t_i) = \mathbf{z}_i, \qquad i = 1, \ldots, N_s, \tag{3.12}$$

$$\mathbf{s}_i^{(n)}(t_{i+1}) = \mathbf{s}_{i+1}^{(n)}(t_{i+1}), \qquad i = 2, \ldots, N_s - 1,\ n = 0, \ldots, d-1. \tag{3.13}$$

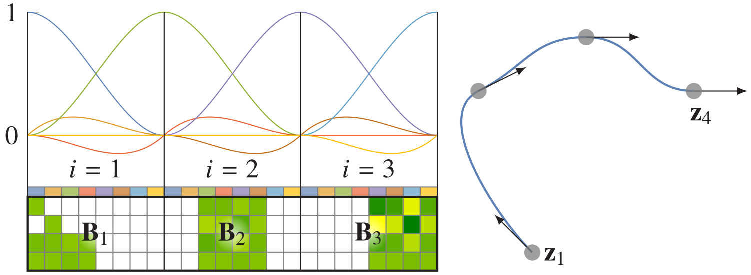


Figure 3.3: Visualization of a Hermite spline of degree three. This spline is compact, and every segment depends only on the neighboring control points and its tangents. Also, all basis functions have the same shape for each segment. On the right, the control points, the tangent vector and the spline curve are shown. The tangent vector length is scaled by 1/3.

Depending on the degree of the spline, additional boundary value conditions are needed. Typical choices specify the derivative of the spline at its beginning and end, which can be expressed as

$$\mathbf{s}_1^{(n)}(t_1) = \left(x_1^{(n)}, y_1^{(n)}\right)^\top, n < d, \tag{3.14}$$

$$\mathbf{s}_{N_s}^{(n)}(t_{N_s+1}) = \left(x_{N_s}^{(n)}, y_{N_s}^{(n)}\right)^\top, n < d. \tag{3.15}$$

This will lead to a linear equation system in $\mathbf{c}_{i,k}$ which has a single solution. The overall spline is of class $C^{(d-1)}$ and passes through the control points. The resulting spline has global support, meaning that changing a single control points affects the whole spline curve. An example is visualized in Fig. 3.2. We selected a degree of three and as boundary condition the second derivative at the beginning and end of the spline to zero.

3.1.2 Hermite Splines

For Hermite splines, the first derivative $\mathbf{z}'_i = (x'_i, y'_i)^\top$ at $(x_i, y_i)^\top$ is given in addition to the measurement points $\mathbf{z}_i = (x_i, y_i)^\top$. $\mathbf{z}$ and $\mathbf{z}'$ are concatenated to form the control points vector $\mathbf{p}_k$ like

$$\mathbf{p}_{2k-1} = \mathbf{z}_k, \tag{3.16}$$

$$\mathbf{p}_{2k} = \mathbf{z}'_k. \tag{3.17}$$

Hermite splines are formulated in the same way as interpolating polynomial splines with the exception that the first derivative is also given. The number of spline segments is $N_s = \frac{N_c}{2} - 1$ and the linear equation system is

$$\mathbf{s}_i(t) = \sum_{k=0}^{d} \mathbf{c}_{i,k+1} t^k, \tag{3.18}$$

$$\mathbf{s}(t_i) = (x_i, y_i), \qquad i = 1, \ldots, N_s + 1, \tag{3.19}$$

$$\mathbf{s}'(t_i) = \left(x'_i, y'_i\right), \qquad i = 1, \ldots, N_s + 1, \tag{3.20}$$

$$\mathbf{s}_i^{(n)}(t_{i+1}) = \mathbf{s}_{i+1}^{(n)}(t_{i+1}), \qquad i = 2, \ldots, N_s,\ n = 0, \ldots, d-2. \tag{3.21}$$

Depending on the degree, one also needs to specify boundary value conditions. A typical choice is presented in the interpolating polynomial spline section in Eq. (3.15). The overall spline is of class $C^{(d-2)}$ resulting directly from Eq. (3.21). A popular choice is a cubic ($d = 3$) Hermite spline which is visualized in Fig. 3.3. This spline does not need any additional boundary value conditions, it has local support, and the shape of each basis function in each segment is the same.

3.1.3 Bézier Splines

Bézier splines are based on Bernstein basis polynomials:

$$\mathrm{b}_k(t) = \binom{d}{k} (1 - t)^{(d-k)} t^k. \tag{3.22}$$

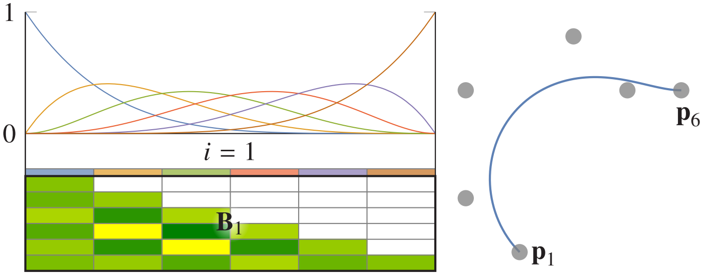


Figure 3.4: Visualization of a Bézier spline using a single spline segment. The spline is based on Bernstein basis polynomials. It is non-compact and approximating as the spline does not pass through all control points.

In this case, the spline is not formulated by solving a linear equation system. Instead, the spline basis functions are modeled directly (see Eq. (3.6)). In general, this leads to an approximating spline which means that the spline does not pass through all control points. We only show a global spline using Bernstein basis polynomials since composite Bézier splines with multiple segments are possible but they are a special case of B-splines discussed in the next section. A Bézier spline for a single spline segment ($N_s = 1$) and with a spline degree of $d = N_c - 1$ is visualized in Fig. 3.4.

3.1.4 B-splines

B-splines are based on B-spline basis polynomials. One way to define them is using the Cox-de Boor recursion formula

$$\mathrm{b}_{i,k}(t) = \mathrm{b}_{i,k,d}(t), \tag{3.23}$$

$$\mathrm{b}_{i,k,j}(t) = \frac{t - \bar{t}_k}{\bar{t}_{k+j} - \bar{t}_k}\mathrm{b}_{i,k,j-1}(t) + \frac{\bar{t}_{k+j+1} - t}{\bar{t}_{k+j+1} - \bar{t}_{k+1}}\mathrm{b}_{i,k+1,j-1}(t), \tag{3.24}$$

$$\mathrm{b}_{i,k,0}(t) = \begin{cases} 1, & \text{if} \quad \bar{t}_k \le t < \bar{t}_{k+1} \\ 0, & \text{otherwise} \end{cases} . \tag{3.25}$$

Due to the construction of its basis, a B-spline has local support as $\mathrm{b}_{i,k,0}$ is only non-zero in $[\bar{t}_k, \bar{t}_{k+1}]$. The number of spline segments is $N_s = N_c - d$.

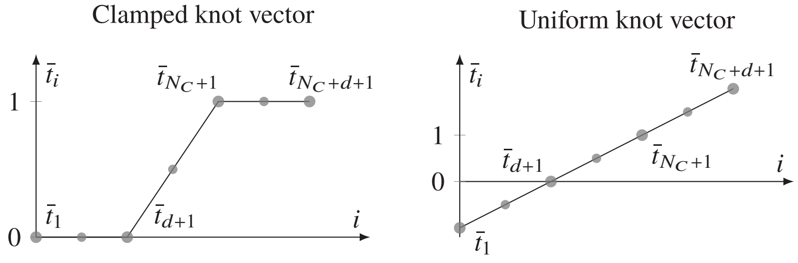


Figure 3.5: Visualization of the clamp and uniform knot vectors.

The vector $\bar{\mathbf{t}} \in \mathbb{R}^{N_c+d+1}$ is called the knot vector which should not be confused with the interval vector $\mathbf{t}$. The knot vector must be monotonically increasing ($\bar{t}_i \leq \bar{t}_{i+1}, \; \forall i$). It controls the differentiability class and thus determines how many control points depend on each segment. We will discuss two commonly used choices: a clamped knot vector and a uniform knot vector.

A clamped knot vector is defined as

$$\bar{t}_i = \begin{cases} 0, & i \leq d+1 \\ \frac{i-1-d}{N_c-d}, & d+1 < i \leq N_c+1 \\ 1, & i > N_c+1 \end{cases} . \tag{3.26}$$

A uniform B-spline is a B-spline for which the spacing of the knot vector is uniform. The knot vector is defined as

$$\bar{t}_i = \frac{i-1-d}{N_c-d} . \tag{3.27}$$

The knot vectors from Eqs. (3.26) and (3.27) are shown in Fig. 3.5.

The basis and the spline curves for both splines are visualized in Fig. 3.6. The main difference between the two knot vectors is that the resulting spline curve of the clamp knot vector starts at the first control point and ends at the last one, whereas the uniform knot vector does not start or end at any control point. From a computational point of view, the uniform B-spline is more efficient, as the basis is the same for each segment and thus can be precomputed. The basis is only dependent on the spline degree.

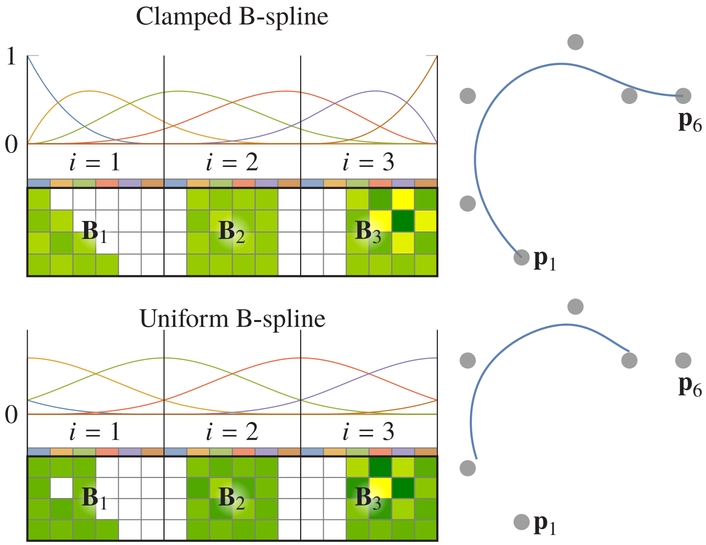


Figure 3.6: Visualization of a clamped and a uniform B-spline. Both have local support. The clamped B-spline starts at the first control point and ends at the last control point. The basis functions of the uniform B-spline are the same for each segment.

3.1.5 Spline Comparison

We summarize the different splines in Table 3.1 by denoting the type of the spline, which is either I for interpolating or A for approximating, the compactness, denoted with yes if the spline is compact (otherwise no), the number of segments N_s and the differentiability class C^k.

As we are mainly interested in modeling a continuous function with splines and optimizing the control points given some measurements, we will introduce estimation algorithms in the next section and then move on to describe which spline basis is beneficial for the use in such estimation algorithms in Section 3.2.7.

Spline	Type	Compact	N_s	C^k
Global polynomial spline	I	no	1	C^∞
Interpolating polynomial spline	I	no	$N_c - 1$	C^{d-1}
Hermite spline	I / A	yes	$\frac{N_c}{2} - 1$	C^{d-2}
Global Bézier spline	A	no	1	C^∞
B-spline	A	yes	$N_c - d$	C^{d-1}

Table 3.1: Comparison of different splines. The spline type is denoted as I for interpolating or A approximating, the compactness is denoted with yes if the spline is compact (otherwise no), the number of segments N_s and the differentiability class C^k

3.2 Estimation

Estimation is the process of finding an approximation to a set of (noisy) measurements. One example is the task of finding a line which best fits a set of given 3D point measurements. To solve such a problem, we first need to select a parametrization of the quantity to estimate (the line, in our example) as well as to model a cost or objective function which describes what ‘best fit’ means (e.g., the sum of distances between each 3D point and the line) and finally to decide on an optimization algorithm which tries to compute the best parameters. We always estimate the parameters by minimization

$$\mathbf{x} = \arg\min_{\mathbf{x}} \mathrm{f_c}(\mathbf{x}), \tag{3.28}$$

where $\mathbf{x}$ are the parameters to be optimized and $\mathrm{f_c}$ is the cost or objective function. This is called a general unconstrained minimization problem.

In this section, we give an introduction to two more specific minimization problems, namely LLS and NLS, as well as describe how rotations and lines in 3D can be parametrized, how problems like estimating the closest point to a set of lines are solved and how splines can be embedded in such minimization problems.

All of these aspects are later used in various chapters for camera calibration like estimating a camera viewing ray given 3D measurements.

3.2.1 Linear Least Squares

The linear least squares (LLS) method is a popular way of modeling cost functions. The cost function has the form

$$\mathrm{f_c}(\mathbf{x}) = \|\hat{\mathbf{y}} - \mathbf{H}\mathbf{x}\|^2. \tag{3.29}$$

Here $\hat{\mathbf{y}}$ are the measurements, $\mathbf{x}$ are the parameters and $\mathbf{H}$ is called the observation matrix. A closed form solution for the optimal parameters of a LLS can be derived by calculating the derivative of Eq. (3.29) with respect to $\mathbf{x}$, setting it to zero and solving the equation for $\mathbf{x}$. This results in

$$\hat{\mathbf{x}} = \left(\mathbf{H}^\top \mathbf{H}\right)^{-1} \mathbf{H}^\top \hat{\mathbf{y}}. \tag{3.30}$$

Numerically, it is often desirable not to compute the inverse explicitly but using a matrix decomposition like singular-value, QR or cholesky decomposition. See, e.g., [Bjö96, Bjö15] for more details about LLS.

If the observation matrix is of full rank, the LLS problem has a single solution which is the global minimum. If $\mathbf{x}$ is over-parametrized, the observation matrix is rank deficient. Such problems can be solved by using a singular-value decomposition (SVD) or a robustified iterative solver if the problem is large and sparse [Ng91].

3.2.2 Non-linear Least Squares

The cost function of a non-linear least squares (NLS) problem has the form

$$\mathrm{f_c}(\mathbf{x}) = \|\mathbf{f}(\mathbf{x})\|^2. \tag{3.31}$$

A NLS problem is more general than a LLS problem which makes it applicable to a wider range of real-world problems, but it has the drawback of not guaranteeing a global optimum. Generally, such problems can only be solved iteratively. This means that given a starting point $\mathbf{x}_1$, the original problem is approximated and a correction $\Delta\mathbf{x}_1$ is determined and added to the starting point in order to get a new starting point $\mathbf{x}_2$. The update step is $\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta\mathbf{x}_i$. Usually, we want to decrease the residual in every step, which means $\|\mathbf{f}(\mathbf{x}_{i+1})\|^2 < \|\mathbf{f}(\mathbf{x}_i)\|^2$, so that every step will take us a little bit closer to the minimum. We call a step a successful step if this criterion is fulfilled, otherwise it is called an unsuccessful step. A new update step is determined until a termination criterion is fulfilled. Termination criteria may be based on the step size as well as the cost function value.

If we approximate the problem by linearizing the cost function and solving the resulting LLS, this results in the Gauss-Newton method:

$$\Delta\mathbf{x}_i = \underset{\Delta\mathbf{x}_i}{\arg\min} \left\| \mathbf{f}(\mathbf{x}_i) + \frac{\partial\mathbf{f}}{\partial\mathbf{x}}(\mathbf{x}_i)\Delta\mathbf{x}_i \right\|^2 . \tag{3.32}$$

In general, this leads to a lot of unsuccessful steps and thus a divergent algorithm. For a stable solver, the step size $\Delta\mathbf{x}_i$ needs to be controlled. The algorithms to control the step size can mainly be divided into two groups: line search and trust region. For all NLS problems we use a Levenberg-Marquardt algorithm, implemented in the Google Ceres Solver[1], which belongs to the trust region group. For a more in-depth description, we refer to [NW06].

Over-parametrized problem formulations can be solved without modifications using a Levenberg-Marquardt algorithm (see [HZ04]), but it is still computationally and numerically more stable to use a so-called local parametrization. A local parametrization $\mathbf{f}_l$ is defined as

$$\mathbf{x}_{i+1} = \mathbf{f}_l(\mathbf{x}_i, \Delta\mathbf{x}_i), \tag{3.33}$$

where $\mathbf{x} \in \mathbb{R}^n$ are the optimization parameters and $\Delta\mathbf{x}_i \in \mathbb{R}^m, m \leq n$ is the controlled update step which needs to be added to the parameters to get the new parameter values for the next iteration. The concept of local parametrizations can be applied even if $\mathbf{x}$ is not over-parametrized as setting $m = n$ and $\mathbf{f}_l(\mathbf{x}_i, \Delta\mathbf{x}_i) = \mathbf{x}_i + \Delta\mathbf{x}_i$ results in a standard parameter update. One property the function $\mathbf{f}_l$ must fulfill is that a step of zero may not alter the parameter vector:

$$\mathbf{x} \overset{!}{=} \mathbf{f}_l(\mathbf{x}, \mathbf{0}), \quad \forall\mathbf{x}. \tag{3.34}$$

[1] `http://ceres-solver.org/` by S. Agarwal, K. Mierle and Others, last retrieved 2020-09-22

Using an over-parametrization is beneficial if the minimal parametrization has singularities or is cumbersome to handle like in the case of rotations or lines. These two cases are covered in more detail in the next two sections.

3.2.3 Parametrization of Rotations in 3D

There are several ways of representing rotations in 3D. In this section, we discuss some of these parametrizations and how a rotation can be represented if used during optimization. We discuss two minimal-parametric representations, Euler angles and Rodrigues vectors, and two non-minimal parametrizations, rotation matrices and unit quaternions. A more detailed description may be found in appendix A6.9 in [HZ04].

Rotation Matrices A matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix if $\mathbf{R}^\top = \mathbf{R}^{-1}$ and $\det(\mathbf{R}) = 1$. Rotating a point $\mathbf{p} \in \mathbb{R}^3$ is simply done by a matrix vector multiplication $\mathbf{p}' = \mathbf{R}\mathbf{p}$. Even though rotation matrices are efficient for transformations, they are not really suitable to be used as optimization parameters due to their high number of parameters and their two constraints which cannot be easily enforced during optimization. Also, the projection step from an arbitrary matrix to a rotation matrix is computationally expensive since it usually involves a SVD or an eigenvalue decomposition (EVD).

Euler Angles Euler angles represent a rotation by three angles α, β, γ. To rotate a point, the Euler angles are usually converted into a rotation matrix $\mathbf{R} = \mathbf{R}(\alpha)\mathbf{R}(\beta)\mathbf{R}(\gamma)$. Euler angles are very intuitive to use but many different conventions exist and they generally suffer from gimbal lock or gauge ambiguity (see [McL00]).

Angle-axis Representation The Rodrigues vector $\mathbf{v}$ represents a rotation with a 3D vector where the rotation angle is defined as $\theta = \|\mathbf{v}\|$ and the normalized rotation axis is $\bar{\mathbf{v}} = {}^{\mathbf{v}}/_{\|\mathbf{v}\|}$. To convert $\mathbf{v}$ to a rotation matrix one can use the exponential map $\mathbf{R} = \exp(\mathrm{skew}(\mathbf{v}))$. The skew function is defined as

$$\mathrm{skew}(\mathbf{v}) = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}. \tag{3.35}$$

When used in a NLS problem, special care needs to be taken for zero degree rotations ($\mathbf{v} = \mathbf{0}$) as their rotation axis is undefined.

Moreover, there is a 2π periodicity as $\mathbf{v}$ and $\mathbf{v} + 2\pi n\bar{\mathbf{v}}, n \in \mathbb{N}$ yields the same rotation matrix. To avoid this ambiguity we can change the length of the vector in steps of 2π until $\|\mathbf{v}\| \in [0, \pi)$:

$$\mathbf{v}^* = \mathbf{v}\left(1 - \frac{2\pi n^*}{\|\mathbf{v}\|}\right), \tag{3.36}$$

$$n^* = \left\lfloor \frac{\|v\| + \pi}{2\pi} \right\rfloor. \tag{3.37}$$

Unit Quaternions A unit quaternion $\mathbf{q} \in \mathbb{H}$ is fully described by a four dimensional vector where $\|\mathbf{q}\| = 1$. We only give a short introduction to quaternions, the interested reader can find more information in, e.g., [Kui99]. Unit quaternions also describe 3D rotations. Compared to Euler angles they do not suffer from gimbal lock, and compared to rotation matrices their composition is more efficient and numerically stable. A unit quaternion can be constructed from a Rodrigues vector as follows (for $\mathbf{v} \neq \mathbf{0}$):

$$\mathbf{q} = \begin{pmatrix} \sin(\theta) \\ \bar{\mathbf{v}}\cos(\theta) \end{pmatrix}. \tag{3.38}$$

To compose unit quaternions, the quaternion product is needed. Since rotations have three degrees of freedom (DOF) but unit quaternions have four parameters, unit quaternions are over-parametrized. We use the local parametrization from appendix A6.9.2 in [HZ04]. The local parametrization $\mathbf{q}^* = \mathbf{f}_{\mathrm{q}}(\mathbf{q}, \Delta\mathbf{q})$ with $\Delta\mathbf{q} \in \mathbb{R}^3$ can be defined as

$$\mathbf{q}^* = \begin{pmatrix} \cos\left(\frac{\|\Delta\mathbf{q}\|}{2}\right) \\ \Delta\mathbf{q}\,\mathrm{si}\left(\frac{\|\Delta\mathbf{q}\|}{2}\right) \end{pmatrix} * \mathbf{q}, \tag{3.39}$$

where $*$ is the quaternion product.

Rotation type	# params.	Local param.	Singularities	Composable
Rotation matrix	9	no	yes	yes
Euler angle	3	N/A	yes	no
Rodrigues vector	3	N/A	yes	no
Unit Quaternions	4	yes	no	yes

Table 3.2: Comparison of the different rotation parametrizations.

Summary

In Table 3.2 we give an overview of the different rotation parametrizations. In this work we will use either Rodrigues vectors or unit quaternions for parametrization of 3D rotation.

3.2.4 Parametrization of Lines in 3D

A line in three dimensional space can be described by a base point $\mathbf{x}_0 \in \mathbb{R}^3$ and a direction $\mathbf{d} \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$. Every point on such a line can be described as

$$\mathbf{x} = \mathbf{x}_0 + s\mathbf{d}, \quad s \in \mathbb{R}. \tag{3.40}$$

A line in 3D is parametrized with six parameters but has 4 DOF. Neither moving the base point in the direction of $\mathbf{d}$ nor changing the norm of the direction vector alter the line. One can add two constraints to eliminate the two extra DOF: The first constraint fixes the norm of the direction vector to one. The second selects the base point to be the point on the line with the shortest distance to the coordinate origin. This can be expressed as

$$\|\mathbf{d}\| = 1, \tag{3.41}$$

$$\mathbf{d} \cdot \mathbf{x}_0 = 0. \tag{3.42}$$

As we have an over-parametrized line, we need a local parametrization (see Section 3.2.2). Hence we seek a function $\mathbf{f}_\text{L} : \mathbb{R}^6 \times \mathbb{R}^4 \to \mathbb{R}^6$ such that we can update a line $\mathbf{l} = (\mathbf{x}_0, \mathbf{d})$ with local coordinates $\Delta\mathbf{y} = (\Delta\mathbf{x}_0, \Delta\mathbf{d})$ with $\Delta\mathbf{x}_0, \Delta\mathbf{d} \in \mathbb{R}^2$ to $\mathbf{l}^* = \mathbf{f}_\text{L}(\mathbf{l}, \Delta\mathbf{y})$. We split the local parametrization into two

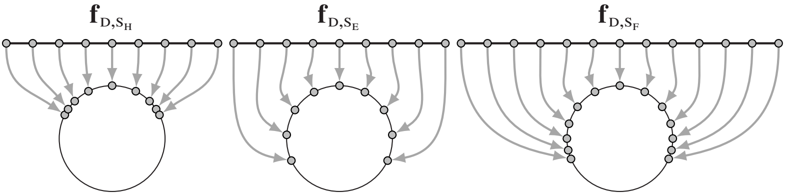

$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{H}}$
$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}$
$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{F}}$

Figure 3.7: Visualization of three mapping functions from two dimensional space to a three dimensional sphere. As the mapping is rotational symmetric only one slice is shown. $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{H}}$ only maps a half sphere, $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}$ maps the whole sphere but has singularities, $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{F}}$ also maps the whole sphere but without singularities. See Eq. (3.44) to Eq. (3.46) for the mathematical formulation.

parts, one for each constraint, and start with the unit length constraint of $\mathbf{d}$ (Eq. (3.41)). This is exactly the parametrization of a 3D sphere with radius one. We follow the n-sphere local parametrization formulation in [HZ04] appendix A6.9.3: The local parametrization $\mathbf{f}_\mathrm{D}$ we seek is

$$\mathbf{d}^* = \mathbf{f}_\mathrm{D}(\mathbf{d}, \Delta\mathbf{d}). \tag{3.43}$$

The update step is split into two parts $\mathbf{d}^* = \mathbf{f}_{\mathrm{D},\mathrm{T}}(\mathbf{d}, \mathbf{f}_{\mathrm{D},\mathrm{S}}(\Delta\mathbf{d}))$, a mapping function $\mathbf{f}_{\mathrm{D},\mathrm{S}} : \mathbb{R}^2 \rightarrow \mathbb{S}^2$ which transforms a cartesian point onto the unit sphere, and a coordinate transformation $\mathbf{f}_{\mathrm{D},\mathrm{T}} : \mathbb{R}^3 \times \mathbb{S}^2 \rightarrow \mathbb{S}^2$. There are several possibilities for $\mathbf{f}_{\mathrm{D},\mathrm{S}}$, three of which are presented here:

$$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{H}}(\Delta\mathbf{d}) = \frac{(\Delta\mathbf{d}, 1)^\top}{\left\|(\Delta\mathbf{d}, 1)^\top\right\|}, \tag{3.44}$$

$$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}(\Delta\mathbf{d}) = \begin{pmatrix} \mathrm{si}(\|\Delta\mathbf{d}\|)\Delta\mathbf{d} \\ \cos(\|\Delta\mathbf{d}\|) \end{pmatrix}, \tag{3.45}$$

$$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{F}}(\Delta\mathbf{d}) = \frac{1}{4 + \|\Delta\mathbf{d}\|^2} \begin{pmatrix} 4\Delta\mathbf{d} \\ 4 - \Delta\mathbf{d} \cdot \Delta\mathbf{d} \end{pmatrix}. \tag{3.46}$$

These functions are visualized in Fig. 3.7. All of these mappings are continuous. $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{H}}$ maps only a half-sphere, $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}$ maps a full sphere but is not bijective. $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{F}}$ also maps the whole sphere excluding one pole, and it is bijective. All of these mappings are well defined at zero and map a zero displacement to $\mathbf{e}_z$.

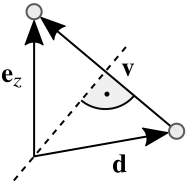


Figure 3.8: Visualization of the Householder transformation. We use it to transform a unit length vector $\mathbf{d}$ to $\mathbf{e}_z$ and vice versa. The dashed line is the reflection plane.

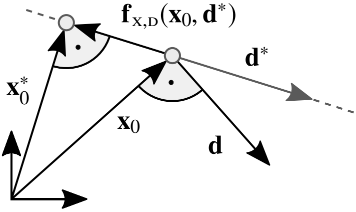


Figure 3.9: Correction of the line base point due to a change of the line direction $\mathbf{f}_{\mathrm{X,D}}(\mathbf{x}_0, \mathbf{d}^*)$.

The next step is to design the coordinate transformation function $\mathbf{f}_{\mathrm{D,T}}$ in such a way that $\mathbf{f}_{\mathrm{D,T}}(\mathbf{d}, \mathbf{f}_{\mathrm{D,S}}(\mathbf{0})) \overset{!}{=} \mathbf{d},\ \forall \mathbf{d}$. The function $\mathbf{f}_{\mathrm{D,T}}$ must be singularity-free and needs to map $\mathbf{e}_z$ to $\mathbf{d}$ since $\mathbf{f}_{\mathrm{D,S}}(\mathbf{0}) = \mathbf{e}_z$. To define such a coordinate transformation, we use a Householder transformation (see, e.g., [VG13] for more information). A Householder transformation describes a reflection on a plane containing the origin. It is a linear transformation and can be described by a Householder matrix $\mathbf{H}$. A Householder matrix is only dependent on the plane normal vector $\mathbf{v} \in \mathbb{R}^3$. We define $\mathbf{f}_{\mathrm{H}} : \mathbb{R}^3 \to \mathbb{R}^{3\times3}, \mathbf{v} \mapsto \mathbf{H}$, as

$$\mathbf{f}_{\mathrm{H}}(\mathbf{v}) = \mathbf{I} - \frac{2}{\mathbf{v}\cdot\mathbf{v}}\mathbf{v}\mathbf{v}^\top. \tag{3.47}$$

See Fig. 3.8 for a visualization of the Householder transformation. The Householder matrix is symmetric ($\mathbf{H} = \mathbf{H}^\top$) and unitary ($\mathbf{H}^\top = \mathbf{H}^{-1}$). As $\mathbf{f}_{\mathrm{D,S}}(\mathbf{d}, \mathbf{0}) = \mathbf{e}_z$, we choose $\mathbf{v}$ in such a way that $\mathbf{e}_z$ is transformed to $\mathbf{d}$ in order to fulfill $\mathbf{f}_{\mathrm{D}}(\mathbf{d}, \mathbf{0}) = \mathbf{d}$. This can be achieved by choosing $\mathbf{v}$ as

$$\mathbf{v} = \mathbf{f}_{\mathrm{V}}(\mathbf{d}) = \begin{cases} \mathbf{d} - \mathbf{e}_z, & d_3 \geq 0 \\ \mathbf{e}_z - \mathbf{d}, & \text{otherwise} \end{cases}. \tag{3.48}$$

The resulting Householder transformation projects the direction vector $\mathbf{d}$ to $\mathbf{e}_z$, which can be written as

$$\mathbf{e}_z = \mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d}))\mathbf{d}. \tag{3.49}$$

As $\mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d}))$ is unitary, we define $\mathbf{f}_{\mathrm{D,T}}$ as

$$\mathbf{f}_{\mathrm{D,T}}(\mathbf{d}, \mathbf{d}_{\boldsymbol{\Delta}}) = \mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d}))\mathbf{d}_{\boldsymbol{\Delta}}. \tag{3.50}$$

The combination of $\mathbf{f}_{\mathrm{D,T}}$ and $\mathbf{f}_{\mathrm{D,S}}$ results in the final local parametrization of a unit vector:

$$\mathbf{d}^* = \mathbf{f}_{\mathrm{D}}(\mathbf{d}, \Delta\mathbf{d}) = \mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d}))\mathbf{f}_{\mathrm{D,S}}(\Delta\mathbf{d}). \tag{3.51}$$

The next step is to define a local parametrization for the base point. This parametrization is a function of the line parameters ($\mathbf{x}_0$, $\mathbf{d}$) and the updates ($\Delta\mathbf{x}_0$, $\Delta\mathbf{d}$)

$$\mathbf{x}_0^* = \mathbf{f}_{\mathrm{X}}(\mathbf{x}_0, \Delta\mathbf{x}_0, \mathbf{d}, \Delta\mathbf{d}). \tag{3.52}$$

We will show two different parametrizations and start with the one that fulfills Eq. (3.42). For modeling, we split that function into two parts, one that copes with the movement of the base point due to a change in the direction $\mathbf{f}_{\mathrm{X,D}}(\mathbf{x}_0, \mathbf{d}^*)$ and the second one that copes with the delta movement $\Delta\mathbf{x}_0$, which we define as $\mathbf{f}_{\mathrm{X},\Delta}(\Delta\mathbf{x}_0, \mathbf{d}^*)$. The final function is the summation of both parts:

$$\mathbf{f}_{\mathrm{X,A}}(\mathbf{x}_0, \Delta\mathbf{x}_0, \mathbf{d}, \Delta\mathbf{d}) = \mathbf{x}_0 + \mathbf{f}_{\mathrm{X,D}}(\mathbf{x}_0, \mathbf{f}_{\mathrm{D}}(\mathbf{d}, \Delta\mathbf{d})) + \mathbf{f}_{\mathrm{X},\Delta}(\Delta\mathbf{x}_0, \mathbf{f}_{\mathrm{D}}(\mathbf{d}, \Delta\mathbf{d})). \tag{3.53}$$

The change of the base point, resulting from a change in the line direction, is computed by moving the base point along the line ($\mathbf{x}_0, \mathbf{d}^*$), until it is the point with the shortest distance to the coordinate origin. This can be achieved by determining the closest point on a plane with respect to $\mathbf{x}_0$, where the plane passes through the coordinate origin and has a normal vector of $\mathbf{d}^*$ (see Fig. 3.9):

$$\mathbf{f}_{\mathrm{X,D}}(\mathbf{x}_0, \mathbf{d}^*) = -(\mathbf{d}^* \cdot \mathbf{x}_0)\mathbf{d}^*. \tag{3.54}$$

The movement of the base according to $\Delta\mathbf{x}_0$ is inspired by the local parametrization of the unit vector. As a tangent plane, we use the plane with the normal vector $\mathbf{d}^*$. To achieve a continuous and singular free mapping, we use a Householder matrix which transforms $\mathbf{d}^*$ to $\mathbf{e}_z$. As only movements perpendicular to $\mathbf{d}^*$ result in a different base point, the base point displacement must be perpendicular to $\mathbf{e}_z$. This is achieved by appending a zero to $\Delta\mathbf{x}_0$. Concatenating these two functions results in

$$\mathbf{f}_{\mathrm{X},\Delta}(\Delta\mathbf{x}_0, \mathbf{d}^*) = \mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d}^*)) \begin{pmatrix} \Delta\mathbf{x}_0 \\ 0 \end{pmatrix}. \tag{3.55}$$

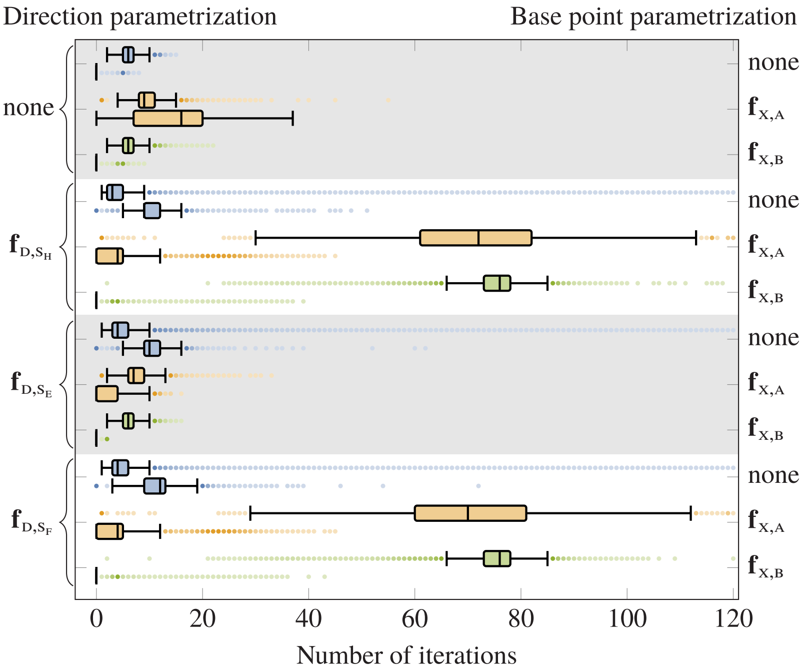


Figure 3.10: Box whisker plot of number of iterations when fitting a line through two points for different line parametrizations. For each combination a box whisker chart of successful and unsuccessful steps is plotted. According to this experiment, the best option is $\mathbf{f}_{\mathrm{D},\mathrm{S_E}}$ combined with $\mathbf{f}_{\mathrm{X,B}}$ as this combination has the lowest number of iterations.

As the Householder matrix is an orthogonal matrix, the second constraint defined in Eq. (3.42) is fulfilled for all $\Delta\mathbf{x}_0$.

A different version of the base point local parametrization may be found if we drop Eq. (3.42) and instead define the movement to be perpendicular only to the original line direction vector $\mathbf{d}$. This results in

$$\mathbf{f}_{\mathrm{X,B}}(\mathbf{x}_0, \Delta\mathbf{x}_0, \mathbf{d}) = \mathbf{x}_0 + \mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d})) \begin{pmatrix} \Delta\mathbf{x}_0 \\ 0 \end{pmatrix}. \tag{3.56}$$

	none	$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{H}}$	$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}$	$\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{F}}$
none	262 (134)	-	-	-
$\mathbf{f}_{\mathrm{X},\mathrm{A}}$	-	726 (6)	21 (0)	738 (12)
$\mathbf{f}_{\mathrm{X},\mathrm{B}}$	-	79 (0)	21 (0)	220 (0)

Table 3.3: Number of iterations needed for convergence when optimizing 10^5 lines jointly. For each line, we generate two point measurements and use the shortest distance between the line and the measurement as the residual. The number of unsuccessful steps is shown in braces, and '-' means no convergence after 1000 steps. $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}$ with $\mathbf{f}_{\mathrm{X},\mathrm{A}}$ or $\mathbf{f}_{\mathrm{X},\mathrm{B}}$ is the best option.

Compared to Eqs. (3.53) to (3.55), this is a simpler local parametrization of the base point as it is not dependent on the updated direction $\mathbf{d}^*$.

To compare the different local line parametrizations, we perform two experiments. The first experiment aims to fit a 3D line through two random points. The initial base point is chosen to be $\mathbf{x}_0 = \mathbf{0}$ and the line direction $\mathbf{d} = \mathbf{e}_z$. The random points are uniformly sampled from a cube with a side length of 200 centered around zero. We solve the NLS problem for a single line where the residuals are the distance vectors between the line and the two points. This experiment is repeated 10^5 times, recording the number of successful and unsuccessful iterations needed for convergence. An unsuccessful step is a step after which the residual is larger than in the previous step. A box whisker plot of the data is shown in Fig. 3.10. As one can see, the half sphere mapping $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{H}}$ (Eq. (3.44)) and the full sphere mapping $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{F}}$ (Eq. (3.46)) require significantly more steps for convergence regardless of the base point local parametrization.

According to this experiment, the best combination is either no local parametrization or $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}$ together with the base point parametrization $\mathbf{f}_{\mathrm{X},\mathrm{B}}$, where the second case has a slightly smaller number of unsuccessful steps.

The second experiment uses the same residuals but optimizes all 10^5 lines in a single optimization problem. Again, the number of successful and the number of unsuccessful iterations are recorded and shown in Table 3.3. The columns of this table show the different line direction parametrizations and the rows show the different base point parametrizations. From this experiment $\mathbf{f}_{\mathrm{D},\mathrm{S}_\mathrm{E}}$ with $\mathbf{f}_{\mathrm{X},\mathrm{A}}$ or $\mathbf{f}_{\mathrm{X},\mathrm{B}}$ is the best option.

We therefore showed that the number of iterations needed for convergence is highly dependent on the choice of local parametrization. The best results in both experiments are achieved by using $\mathbf{f}_{\mathrm{D,S_E}}$ together with $\mathbf{f}_{\mathrm{X,A}}$ or $\mathbf{f}_{\mathrm{X,B}}$.

So in this work, we will use $\mathbf{f}_{\mathrm{D,S_E}}$ as a line parametrization for the directional part as it always results in the lowest number of iterations and $\mathbf{f}_{\mathrm{X,B}}$ for the base point as it is simpler and thus faster to compute than $\mathbf{f}_{\mathrm{X,A}}$.

To summarize, the used local line parametrization is

$$\mathbf{d}^* = \mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d})) \begin{pmatrix} \mathrm{si}(\|\Delta\mathbf{d}\|)\Delta\mathbf{d} \\ \cos(\|\Delta\mathbf{d}\|) \end{pmatrix}, \tag{3.57}$$

$$\mathbf{x}_0^* = \mathbf{x}_0 + \mathbf{f}_{\mathrm{H}}(\mathbf{f}_{\mathrm{V}}(\mathbf{d})) \begin{pmatrix} \Delta\mathbf{x}_0 \\ 0 \end{pmatrix}. \tag{3.58}$$

In the following two Sections 3.2.5 and 3.2.6, we discuss two specific estimation problems which can be solved in closed form. We will then finish the estimation Section 3.2 by demonstrating how spline functions can be integrated into a least squares (LS) problem (Section 3.2.7).

3.2.5 Estimate Line given 3D Points

In this section we derive how to estimate a line given a set of 3D points. We model the line as

$$\mathbf{x} = \mathbf{x}_0 + s\mathbf{d}, \quad s \in \mathbb{R}, \tag{3.59}$$

where $\mathbf{x}_0 \in \mathbb{R}^3$ is the base point and $\mathbf{d} \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$ the direction vector, and denote the set of N points as $\mathbf{p}_i \in \mathbb{R}^3$, where $i = 1, \ldots, N$. The line should have the smallest possible distance to all points, so as the cost function we use the sum of the shortest distances between the line and the points is used. We express the cost function as

$$\sum_{i=1}^{N} \|\mathbf{e}_r\|^2 \tag{3.60}$$

with

$$\mathbf{e}_r = \mathbf{\Delta}_i - \left(\mathbf{\Delta}_i \cdot \overline{\mathbf{d}}\right)\overline{\mathbf{d}}, \tag{3.61}$$

$$\mathbf{\Delta}_i = \mathbf{p}_i - \mathbf{x}_0, \tag{3.62}$$

$$\overline{\mathbf{d}} = \frac{\mathbf{d}}{||\mathbf{d}||}. \tag{3.63}$$

The best fit base point is the mean of all measurements:

$$\hat{\mathbf{x}}_0 = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i. \tag{3.64}$$

To calculate the directional part $\hat{\mathbf{d}}$, a principal component analysis (PCA) may be used. The directional part is the first principal component which means it has the largest possible variance. To apply the PCA, we need the data matrix with column-wise zero empirical mean $\mathbf{X} \in \mathbb{R}^{N \times 3}$ which in this case is

$$\mathbf{X} = \begin{pmatrix} \mathbf{p}_1 - \hat{\mathbf{x}}_0 \\ \vdots \\ \mathbf{p}_N - \hat{\mathbf{x}}_0 \end{pmatrix}. \tag{3.65}$$

Finally, the directional part can either be calculated using an EVD of $\mathbf{X}^T\mathbf{X}$ or via a SVD of $\mathbf{X}$. For more details about PCA see, e.g., [Shl14].

3.2.6 Estimate Closest Point to a Set of Lines

In this section we seek the closest point to a set of lines. We denote the point as $\mathbf{p}$ and use the sum of squared distances to the N given lines as the cost function.

Starting from the line in Eq. (3.59) and the point to line distance in Eqs. (3.61) to (3.63) the following relationship holds:

$$\begin{aligned}
& & \mathbf{p} - \mathbf{x}_{0,i} &= \Big(\big(\mathbf{p} - \mathbf{x}_{0,i}\big) \cdot \overline{\mathbf{d}}_i\Big)\overline{\mathbf{d}}_i \\
\Leftrightarrow & & \Big(\mathbf{x_0} \cdot \overline{\mathbf{d}}_i\Big)\overline{\mathbf{d}}_i - \mathbf{x}_{0,i} &= \Big(\mathbf{p} \cdot \overline{\mathbf{d}}_i\Big)\overline{\mathbf{d}}_i - \mathbf{p} \\
\Leftrightarrow & & \underbrace{\Big(\mathbf{x_0} \cdot \overline{\mathbf{d}}_i\Big)\overline{\mathbf{d}}_i - \mathbf{x}_{0,i}}_{\mathbf{x}_i} &= \underbrace{\Big(\overline{\mathbf{d}}_i^\top \overline{\mathbf{d}}_i - \mathbf{I}\Big)}_{\mathbf{H}_i}\mathbf{p} \\
\Leftrightarrow & & \mathbf{x}_i &= \mathbf{H}_i \mathbf{p}. \qquad (3.66)
\end{aligned}$$

The resulting equation is linear. So this is a LLS problem with three residuals per line.

3.2.7 Spline Functions in Least Squares Problems

Splines are a flexible way of modeling continuous functions, as the fidelity by the number of control points and the differentiability class by the spline degree can both be controlled in an intuitive manner. Once having selected the spline degree and the number of control points, we are left with the task of estimating the control point positions. Given some measurements, we estimate the control points by formulating and than solving a LS problem. To increase the robustness of the estimation, we would like to also integrate a prior in the problem formulation. In this section, we assess what the requirements are for a spline basis so that it can be integrated into a LS problem efficiently and how a spline prior can be formulated.

Spline Basis

As seen in the spline summary (Section 3.1.5), splines can be either interpolating or approximating and either compact or non-compact. A non-compact spline with many control points will result in a huge, dense LS problem which is very slow to solve. The property of approximating or interpolating is usually not important when the control points are estimated by solving a minimization problem.

In this work, we choose the uniform B-spline basis, as this results in a spline which is compact and whose differentiability class can be controlled by the spline degree. Also, the uniform B-spline basis is efficient to compute, as the spline basis is the same for each segment and only depends on the spline degree d. Using a uniform B-spline basis we can simplify the piecewise function definition of Eq. (3.3) for each segment by a single spline function $\mathbf{s}\colon \mathbb{R} \to \mathbb{R}^{N_o}$

$$\mathbf{s}(t) = \mathbf{P}_{\mathrm{i_s}(t)} \mathbf{B}_d^\top \mathbf{a}(\mathrm{i_v}(t)), \tag{3.67}$$

where $\mathbf{B}_d$ is the uniform B-spline basis matrix, $\mathbf{P}$ the control points matrix and $\mathbf{a}(t)$ the polynomial vector (see Eq. (3.7)). The segment function $\mathrm{i_s}: \mathbb{R} \to \mathbb{N}$ and the value function $\mathrm{i_v}: \mathbb{R} \to \mathbb{R}$ are defined as

$$\mathrm{i_s}(t) = \lfloor t \cdot (N_c - d) \rfloor, \tag{3.68}$$

$$\mathrm{i_v}(t) = t \cdot (N_c - d) - \mathrm{i_s}(t), \tag{3.69}$$

where N_c is the total number of control points. The control point matrix is defined as

$$\mathbf{P}_{\mathrm{i_s}(t)} = (\mathbf{p}_{\mathrm{i_s}(t)+1}, \mathbf{p}_{\mathrm{i_s}(t)+2}, \ldots, \mathbf{p}_{\mathrm{i_s}(t)+d+1}). \tag{3.70}$$

In case of a spline surface, we get $\mathbf{s}\colon \mathbb{R}^2 \to \mathbb{R}^{N_o}$:

$$\mathbf{s}(u, v) = \sum_{k=1}^{d+1} \sum_{l=1}^{d+1} \mathrm{b}_k(u) \mathrm{b}_l(v) \mathbf{p}_{\mathrm{i_s}(v)+l, \mathrm{i_s}(u)+k}, \tag{3.71}$$

$$\mathbf{b}(t) = \mathbf{B}_d^\top \mathbf{a}(\mathrm{i_v}(t)). \tag{3.72}$$

The control points are now laid out on a two dimensional grid. The basis itself is separable and depends only on one variable.

If $N_o = 1$, this can be written in matrix notation without the need for tensors:

$$s(u, v) = \mathbf{a}^\top(\mathrm{i_v}(v)) \mathbf{B}_d \mathbf{P}_{\mathbf{c}, \mathrm{i_s}(v), \mathrm{i_s}(u)} \mathbf{B}_d^\top \mathbf{a}(\mathrm{i_v}(u)), \tag{3.73}$$

$$\mathbf{P}_{\mathbf{c},i,j} = \begin{pmatrix} p_{i+1,j+1} & \cdots & p_{i+1,j+d+1} \\ \vdots & \ddots & \vdots \\ p_{i+d+1,j+1} & \cdots & p_{i+d+1,j+d+1} \end{pmatrix}. \tag{3.74}$$

For the problems we are solving, the position (u, v) at which the spline is evaluated is fixed during optimization, and only the control point positions are estimated. Given a known position, the spline evaluation during optimization is reduced to a single matrix vector product. This makes it really efficient in computing the spline value during optimization.

Smoothness

If one uses splines in a NLS problem, it is often desirable to use a prior. This usually helps in reducing the number of iterations and reduces the effect of outliers. We formulate a prior which is dependent only on the control points. The prior we choose is that we want to have a smooth function. For a spline $\mathbf{s}\colon \mathbb{R} \to \mathbb{R}^{N_o}$ this can be formulated as

$$e_s = \int_0^1 \left\| \mathbf{s}^{(n)}(t) \right\|^2 \mathrm{dt}. \tag{3.75}$$

Using e_s in a NLS problem means minimizing the absolute integral of the n$^{\mathrm{th}}$ derivative.

When minimizing e_s, $\mathbf{s}^{(n)}(t)$ will be zero for all t. Intuitively, the optimization result for $n = 0$ is a null spline, for $n = 1$ a constant spline and for $n = 2$ a straight line.

The question which arises is how to integrate such a smoothness term into a NLS problem. A straightforward way would be to numerically integrate the function using Gaussian quadrature

$$e_s = \int_0^1 \left\| \mathbf{s}^{(n)}(t) \right\|^2 \mathrm{dt} \approx \sum_i \left\| \sqrt{w_i} \mathbf{s}^{(n)}(t_i) \right\|^2, \tag{3.76}$$

where $w_i > 0$ are the integration weights. But in the case of a 1D and 2D uniform B-spline the smoothness integral can be calculated in closed form. We start with the smoothness derivation of a 1D uniform B-spline followed by the 2D uniform B-spline.

For the 1D uniform B-spline smoothness we need to find the relationship

$$e_s = \int_0^1 \left\| \mathbf{s}^{(n)}(t) \right\|^2 \mathrm{dt} \overset{?}{=} ||\mathbf{f}_{\mathrm{S}}(\mathbf{P})||_{\mathrm{F}}^2. \tag{3.77}$$

Using the Frobenius norm of the matrix makes it possible to integrate it directly into a LS problem.

Without loss of generality, the following proof is shown for a single spline segment as every segment is similar for a uniform B-spline. We dropped the indices and write $\mathbf{a}(t)$ instead of $\mathbf{a}(i_v(t))$ and to increase readability.

$$\begin{aligned}\int_0^1 \left\|\mathbf{s}^{(n)}(t)\right\|^2 \mathrm{dt} &= \int_0^1 \mathbf{s}^{\top(n)}(t)\mathbf{s}^{(n)}(t)\,\mathrm{dt} &(3.78)\\ &= \int_0^1 \mathbf{a}^{\top(n)}(t)\mathbf{B}_d\mathbf{P}^\top\mathbf{P}\mathbf{B}_d^\top\mathbf{a}^{(n)}(t)\,\mathrm{dt} &(3.79)\\ &= \mathrm{Tr}\left(\mathbf{P}\mathbf{B}_d^\top \underbrace{\int_0^1 \mathbf{a}^{(n)}(t)\mathbf{a}^{\top(n)}(t)\,\mathrm{dt}\,\mathbf{B}_d}_{\mathbf{S}}\mathbf{P}^\top\right) &(3.80)\\ &= \left\|\mathbf{P}\mathbf{B}_d^\top\mathbf{S}^{1/2}\right\|_{\mathrm{F}}^2. &(3.81)\end{aligned}$$

So this leads to

$$\begin{aligned}\mathbf{f}_{\mathrm{S}}(\mathbf{P}) &= \mathbf{P}\mathbf{B}_s, &(3.82)\\ \mathbf{B}_s &= \mathbf{B}_d^\top\left(\int_0^1 \mathbf{a}^{(n)}(t)\mathbf{a}^{\top(n)}(t)\,\mathrm{dt}\right)^{\frac{1}{2}}. &(3.83)\end{aligned}$$

We see that $\mathbf{B}_s$ only depends on the smoothness derivative n and the spline degree d. This means that $\mathbf{B}_s$ can be precomputed which makes it really efficient to compute it in a LS problem.

Now we extend the above formulation to a 2D uniform B-spline $\mathbf{s}\colon \mathbb{R}^2 \to \mathbb{R}^{N_o}$. Without loss of generality and similar to the proof of the 1D case, the following proof is shown for a single spline segment, $N_o = 1$, $n = 1$, $\mathbf{P_c} = \mathbf{P}_{\mathbf{c},0,0}$ and $\mathbf{Z} = \mathbf{B}_d \mathbf{P_c} \mathbf{B}_d^\top$:

$$\int_0^1 \int_0^1 \|\mathrm{s}(u,v)\|^2 \,\mathrm{du}\,\mathrm{dv} = \int_0^1 \int_0^1 \mathbf{a}^\top(u)\mathbf{Z}^\top \mathbf{a}(v)\mathbf{a}^\top(v)\mathbf{Z}\mathbf{a}(u)\,\mathrm{du}\,\mathrm{dv} \tag{3.84}$$

$$= \int_0^1 \mathbf{a}^\top(u)\mathbf{Z}^\top \underbrace{\int_0^1 \mathbf{a}(v)\mathbf{a}^\top(v)\,\mathrm{dv}}_{\mathbf{S}}\ \mathbf{Z}\mathbf{a}(u)\,\mathrm{du} \tag{3.85}$$

$$= \mathrm{Tr}\left(\mathbf{Z}^\top \mathbf{S}\mathbf{Z} \underbrace{\int_0^1 \mathbf{a}(u)\mathbf{a}^\top(u)\,\mathrm{du}}_{\mathbf{S}}\right) \tag{3.86}$$

$$= \mathrm{Tr}\left(\mathbf{S}^{1/2}\mathbf{Z}^\top \mathbf{S}^{1/2}\mathbf{S}^{1/2}\mathbf{Z}\mathbf{S}^{1/2}\right) \tag{3.87}$$

$$= \left\|\mathbf{S}^{1/2}\mathbf{Z}\mathbf{S}^{1/2}\right\|_\mathrm{F}^2 \tag{3.88}$$

$$= \|\underbrace{\mathbf{B}_s^\top \mathbf{P_c}\mathbf{B}_s}_{\mathbf{f}_\mathrm{S}(\mathbf{P_c})}\|_\mathrm{F}^2. \tag{3.89}$$

For $n = 2$, this results in the smoothing thin plate spline [Duc77].

We have not found a way to extend this formula further for higher dimensional uniform B-splines, but as we will only deal with spline surfaces this is enough for us.

4 Camera Models

Camera models describe mappings between a 2D plane and a 3D world. In computer vision the 2D plane is an image sensor, and the 3D world is the environment which one wants to observe. Typically, all commonly used cameras need a lens to map the 3D world to a 2D plane. The lens is the actual device which is mounted in front of the image sensor.

In this chapter, different mathematical models of lenses are introduced. We call them camera models. "A camera model is fully defined by either using a forward or backward model. The forward model is defined as $\mathcal{P}_\mathrm{F} : \mathbb{R}^3 \to \mathbb{R}^2$, which projects a 3D point in space onto the two-dimensional image space. The backward model is defined as $\mathcal{P}_\mathrm{B} : \mathbb{R}^2 \to \mathbb{P}^3$ which maps a two-dimensional image point to a line in $\mathbb{R}^3$ which is called viewing ray." [2]

In the following sections, the forward model is usually parametrized as $\mathbf{p} \mapsto \mathbf{u}$ and the backward model as $\mathbf{u} \mapsto (\mathbf{x}_0, \mathbf{d})$, where $\mathbf{p} \in \mathbb{R}^3$ is a point in 3D, $\mathbf{u} \in \mathbb{R}^2$ is a point in image space, $\mathbf{x}_0 \in \mathbb{R}^3$ is the base point and $\mathbf{d} \in \mathbb{R}^3$ the direction of a line in 3D.

Usually, we use the backward model to describe the camera model, as oftentimes the forward model cannot be expressed in closed form. We divide the backward model into the directional part $\mathbf{d} = \mathcal{P}_{\mathrm{B,D}}(\mathbf{u})$ and the base point part $\mathbf{x}_0 = \mathcal{P}_{\mathrm{B,X}}(\mathbf{u})$ (see Fig. 4.1). The simplest choice of $\mathcal{P}_{\mathrm{B,X}}$ is the null function. This means that all viewing rays pass through a single point. Such models are called central or sometimes single view point models. For all other non-trivial choices of $\mathcal{P}_{\mathrm{B,X}}$, the models are called non-central or non-single viewpoint models. The parameters which define a camera model are called its intrinsic parameters and are denoted as $\mathbf{k}_\mathrm{I}$.

There are several ways how a camera model may be mathematically formulated. Most models can be classified as global, discrete or local camera models. We will describe a generic global camera model, followed by the discrete camera model and the novel B-spline distortion camera model belonging to the class of

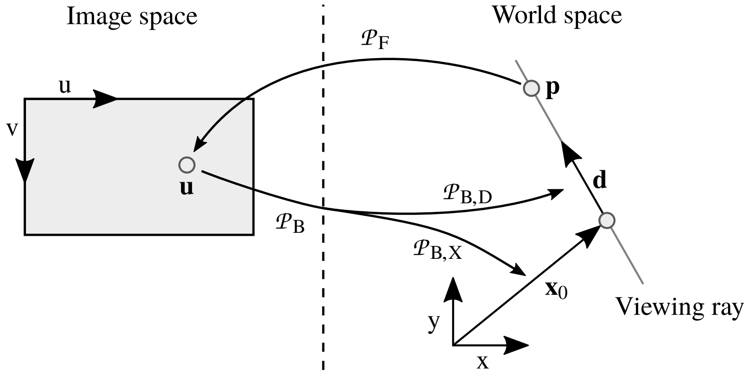


Figure 4.1: Visualization of the camera model function. The forward projection $\mathcal{P}_\mathrm{F}$ transforms a world point $\mathbf{p}$ to image space coordinates $\mathbf{u}$, and the backward projection $\mathcal{P}_\mathrm{B}$ transforms an image point $\mathbf{u}$ to a viewing ray which is modeled as a base point $\mathbf{x}_0$ and a direction vector $\mathbf{d}$.

local camera models. We finish the chapter by showing how the Jacobian matrix of an implicit forward camera model can be derived and how a windshield in front of a camera influences the viewing rays.

4.1 Global Camera Model

A global camera model is usually defined by only a few parameters, which affect the mapping globally. This means a change in one parameter varies the viewing rays of every pixel. These camera models are widely used, as they are relatively easy to calibrate due to their low number of parameters resulting from a usually physically motivated modeling.

We will start with the pinhole camera model without or with distortions by specifying the forward model. Then more general camera models are described by formulating the backward projection $\mathcal{P}_\mathrm{B}$ separated into the directional part $\mathcal{P}_\mathrm{B,D}$ and the base point part $\mathcal{P}_\mathrm{B,X}$.

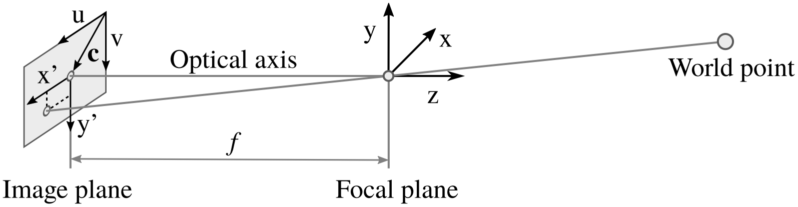


Figure 4.2: The projection model of an ideal pinhole camera model. This model is fully defined by the focal length f and the principal point $\mathbf{c}$.

4.1.1 Viewing Ray Direction Modeling

Pinhole Camera Model

A pinhole camera model is the classic camera model used to project 3D worlds to an image. The appearance of such an image is very natural due to the fact that straight lines in 3D remain straight lines in the image space. The ideal pinhole camera model can be described by three parameters: the focal length f and the principal point $\mathbf{c} = (c_u, c_v)^\top \in \mathbb{R}^2$ (see Fig. 4.2). The focal length is the distance between the focal plane and the image plane. It is usually specified in pixel or metric coordinates. These values can be converted using the pixel pitch of the image sensor. The focal length is directly related to the FOV. A larger focal length results in a smaller FOV and vice versa. The principal point is the orthogonal shift of the image plane coordinate system with respect to the optical axis.

The backward model can be derived geometrically from Fig. 4.2:

$$\mathcal{P}_{\mathrm{B,D}}(u, v) = \begin{pmatrix} u - c_u \\ v - c_v \\ f \end{pmatrix}. \tag{4.1}$$

In the case of an ideal pinhole camera model, the forward model has a closed form and can be written as

$$\mathcal{P}_{\mathrm{F}}(x, y, z) = \begin{pmatrix} f\frac{x}{z} + c_u \\ f\frac{y}{z} + c_v \end{pmatrix}. \tag{4.2}$$

In practice, it is not possible to produce a lens which can be described by the ideal pinhole camera model due to costs, size constraints and manufacturing accuracy limitations. In order to compensate for these kind of errors, the ideal camera model is extended by so-called distortion coefficients. Distortions can be separated into a radial function $\mathrm{f}_\mathrm{R} \colon \mathbb{R} \rightarrow \mathbb{R}$ and a tangential function $\mathbf{f}_\mathrm{T} \colon \mathbb{R}^3 \rightarrow \mathbb{R}^3$. They are embedded into the camera model as

$$\mathbf{u}_\mathrm{I} = \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}, \tag{4.3}$$

$$\mathbf{u}_\mathrm{D} = \mathbf{u}_\mathrm{I}\mathrm{f}_\mathrm{R}(\|\mathbf{u}_\mathrm{I}\|) + \mathbf{f}_\mathrm{T}(\mathbf{u}_\mathrm{I}), \tag{4.4}$$

$$\mathcal{P}_\mathrm{F}(x, y, z) = f\mathbf{u}_\mathrm{D} + \mathbf{c}. \tag{4.5}$$

In literature, a lot of different distortion models are proposed. One of the most popular choices is [WCH92]

$$\mathrm{f}_\mathrm{R}(r) = 1 + \sum_{i=1}^{N_r} k_i r^{2i}, \tag{4.6}$$

$$\mathbf{f}_\mathrm{T}(u, v) = \begin{pmatrix} 2p_1uv + p_2(r^2 + 2u^2) \\ p_1(r^2 + 2v^2) + 2p_2uv \end{pmatrix}, \tag{4.7}$$

$$r = \|\mathbf{u}_\mathrm{I}\|, \tag{4.8}$$

with the radial distortion parameters k_i and the tangential distortion parameters p_1 and p_2. Usually, all odd powers of the radial distortion polynomial are dropped to preserve circular symmetry.

In general, it is sufficient to choose $N_r = 3$ which will result in a model with eight intrinsic parameters:

$$\mathbf{k}_\mathrm{I,G} = (f, c_u, c_v, k_1, k_2, k_3, p_1, p_2)^\top. \tag{4.9}$$

Generalized Camera Model

The pinhole camera model cannot describe cameras with a FOV ≥ 180°. For such lenses, a generalized central camera model is usually formulated in backward direction. This makes it possible to later be generalized into a non-central camera model. In the following, we will describe the camera model proposed in [Str15] that fits the above requirements.

In this model, the direction vector is described using spherical coordinates with an azimuthal angle θ and the polar angle ϕ as

$$\mathbf{d}(\theta, \phi) = \begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}. \tag{4.10}$$

Now we seek a function which maps an image point to the azimuthal and the polar angle. This function is split into two parts, an ideal camera model part and a distortion part. The distortion part is similar to Eqs. (4.6) to (4.8) of the pinhole camera model, but it points in the opposite direction:

$$\mathbf{u}_\mathrm{I} = \frac{1}{f}(\mathbf{u} - \mathbf{c}), \tag{4.11}$$

$$\mathbf{u}_\mathrm{D} = \mathbf{u}_\mathrm{I}\mathrm{f}_\mathrm{R}(\|\mathbf{u}_\mathrm{I}\|) + \mathbf{f}_\mathrm{T}(\mathbf{u}_\mathrm{I}). \tag{4.12}$$

The ideal part is defined as

$$\phi = \arctan\left(\frac{u_{\mathrm{D},2}}{u_{\mathrm{D},1}}\right), \tag{4.13}$$

$$\theta = \mathrm{f}_\mathrm{P}^{-1}(\|\mathbf{u}_\mathrm{D}\|), \tag{4.14}$$

where $\mathrm{f}_\mathrm{P} : \mathbb{R} \to \mathbb{R}, \theta \mapsto r$. The azimuthal angle θ depends on the used lens. In literature, mainly the following formulations are used for f_P [KB06]:

Perspective projection $\quad \mathrm{f}_\mathrm{P}(\theta) = \tan(\theta) \quad$ (4.15)

Stereographic projection $\quad \mathrm{f}_\mathrm{P}(\theta) = 2\tan\left(\frac{\theta}{2}\right) \quad$ (4.16)

Equidistance projection $\quad \mathrm{f}_\mathrm{P}(\theta) = \theta \quad$ (4.17)

Equisolid angle projection $\quad \mathrm{f}_\mathrm{P}(\theta) = 2\sin\left(\frac{\theta}{2}\right) \quad$ (4.18)

Orthogonal projection $\quad \mathrm{f}_\mathrm{P}(\theta) = \sin(\theta) \quad$ (4.19)

The directional part of the camera model is

$$\mathcal{P}_{\mathrm{B,D}}(\mathbf{u}) = \mathbf{d}(\theta, \phi), \tag{4.20}$$

where ϕ and θ is from Eqs. (4.13) and (4.14) respectively.

This camera model has the same number of intrinsic parameters as the standard pinhole camera model. When choosing f_P as the perspective projection, the resulting model describes a pinhole camera model with distortions formulated in backward direction.

4.1.2 Base Point Modeling

In general, the non-central part of the camera model is often neglected for pinhole and even for fisheye lenses and set to zero. But if targets like checkerboards with little distance to the camera are used for calibration, the central camera model approximation will induce errors in the directional part of the camera model.

As described in [SRT+11], there are mainly two kinds of non-central cameras: In the so-called oblique model, all viewing rays are skew to each other. In the axial or x-slit model, all rays hit a single or two axis. For fisheye cameras, an axial model is usually sufficient (see [Gen06]).

In axial models, the base point function $\mathcal{P}_{\mathrm{B,X}}$ only has a z-component, and for fisheye lenses, this component is radially symmetric. We formulate this part as

$$\mathcal{P}_{\mathrm{B,X}}(\mathbf{u}) = \begin{pmatrix} 0 \\ 0 \\ \mathrm{f}_\mathrm{z}(\theta) \end{pmatrix}, \tag{4.21}$$

where θ is from Eq. (4.14).

As shown in [Gen06], the base point of the viewing rays can be described as

$$\mathrm{f}_\mathrm{z}(\theta) = \left(\frac{\theta}{\sin(\theta)} - 1\right) \sum_{i=0}^{N_D} \epsilon_i \theta^{2i}, \tag{4.22}$$

where $\epsilon_i, i = 1, \ldots, N_D$, are the intrinsic base point parameters.

We use $N_D = 2$ which adds three additional parameters to the intrinsic parameters, namely

$$\mathbf{k}_{\mathrm{I,B}} = (\epsilon_0, \epsilon_1, \epsilon_2)^\top. \tag{4.23}$$

4.2 Discrete Camera Models

Discrete camera models estimate the mapping between a pixel coordinate and a viewing ray at discrete points, e.g., at the center of every pixel. This means we need at least two 3D measurements for each ray. This can be achieved by interpolating sparse measurements [SR04] or by using a structured light technique [GN05]. In addition, not only the geometric properties of the ray can be estimated but also the radiometric properties and the point spread function. Such a viewing ray is called 'raxel' [GN05]. Since this work focuses only on geometric camera calibration, the intrinsic parameters are

$$\mathbf{k}_{\mathrm{I,R}} = \left(\mathbf{x}_{0,1}, \mathbf{d}_1, \mathbf{x}_{0,2}, \mathbf{d}_2, \ldots\right). \tag{4.24}$$

The viewing ray is denoted by

$$\mathbf{x} = \mathbf{x}_{0,i} + s\mathbf{d}_i, s \in \mathbb{R}. \tag{4.25}$$

If a continuous camera model is needed for steps like undistortion, the viewing rays can be interpolated or fitted to a local camera model (see Section 5.1.4).

4.3 Generic B-spline Distortion Camera Model

In this section, we describe the novel generic B-spline distortion camera model. The goal of this camera model is to design a model which is more generic than a global model (Section 4.1) and still continuous which is not the case for discrete camera models (Section 4.2).

The use of uniform B-splines for this camera model gives us control over the number of intrinsic parameters determined by the number of control points as well as the smoothness determined by the spline degree. Uniform B-splines are efficient to compute due to the uniform basis and the local support. Also, B-splines can be efficiently integrated into NLS problems (see Section 3.2.7).

The question is how to formulate a camera model using splines. For the directional part, it is possible to use a B-spline which maps an image point to a 3D vector (see [RW12] and [2]). Similarly, for the support point a spline mapping from an image point to a 3D point can be used (see [RW12, SLPS20]), with the drawback that the use of an over-parametrized spline function leads to more parameters (six instead of four for the full model). Also, a prior cannot easily be integrated.

We propose a B-spline distortion model which is minimally parametric in both the direction part and the base point part.

4.3.1 Viewing Ray Direction Modeling

Instead of modeling the viewing ray direction directly by a spline, we use the spline as an image distortion spline which maps an image point to a distorted image point. The distorted image point is afterwards mapped to a unit sphere which describes the viewing ray direction.

To map the distorted image point to a unit sphere, we use the ideal part of the formulation of the generalized global camera model (Eq. (4.13) to Eq. (4.20)). The choice of the projection model does not need to match the type of the lens as the distortion function is generic enough to model the remaining deviations. Since we would like to handle lenses with a FOV > 180°, we use the equidistance projection function. This leads to

$$\mathcal{P}_{\mathrm{B,D}}(u,v) = \begin{pmatrix} \sin(r)\cos(\arctan(c)) \\ \sin(r)\sin(\arctan(c)) \\ \cos(r) \end{pmatrix}, \tag{4.26}$$

$$r = \|\mathbf{f}_{\mathrm{D}}(u,v)\|, \tag{4.27}$$

$$c = \frac{\mathrm{f}_{\mathrm{D},y}(u,v)}{\mathrm{f}_{\mathrm{D},x}(u,v)}, \tag{4.28}$$

where $\mathbf{f}_{\mathrm{D}} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is the generic distortion function. The principal point and the focal length is no longer modeled explicitly but they are contained in the generic distortion function.

This formulation can be simplified by using the relationships

$$\sin(\arctan(c)) = \frac{c}{\sqrt{1+c^2}}, \tag{4.29}$$

$$\cos(\arctan(c)) = \frac{1}{\sqrt{1+c^2}}, \tag{4.30}$$

which results in

$$\mathcal{P}_{\mathrm{B,D}}(u,v) = \begin{pmatrix} \mathbf{f}_{\mathrm{D}}(u,v)\,\mathrm{si}(r) \\ \cos(r) \end{pmatrix}, \tag{4.31}$$

where $\mathrm{si}(r) = \frac{\sin(r)}{r}$.

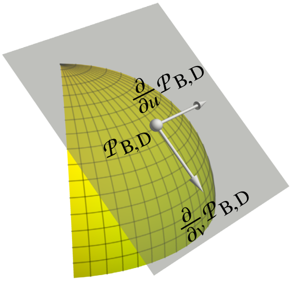
For an ideal pinhole camera model, the distortion function $\mathbf{f}_{\mathrm{D}}$ is a plane (see Eq. (4.11)). Therefore, we use a plane as a prior for camera calibration. This can be efficiently integrated into a NLS as we can directly use the smoothness of the third spline derivative (see Section 3.2.7).

4.3.2 Base Point Modeling

In literature, the non-central part of a camera model is mainly studied for fisheye or catadioptric lenses since the modeling of the non-central part is usually physically motivated. For catadioptric lenses, the mirror surface is parametrized and estimated, and for fisheye lenses a model of the caustic is used. These kinds of models cannot describe non-central rays introduced by, e.g., a windshield which is placed in front of a lens (see Section 4.5). We propose a more general approach to model the base point of viewing rays using uniform B-splines.

We observe that the base point has only two degrees of freedom as moving the base point along the line direction results in the same ray. This means only the displacement in the plane with a normal vector parallel to the viewing ray direction results in a different ray. Therefore, the description of the non-central part is formulated in the tangent space of $\mathcal{P}_{\mathrm{B,D}}$ (see Fig. 4.3). We use the derivatives of $\mathcal{P}_{\mathrm{B,D}}$ as a curvilinear basis and describe the base point as

$$\mathcal{P}_{\mathrm{B,X}}(\mathbf{u}) = \frac{\partial \mathcal{P}_{\mathrm{B,D}}}{\partial \mathbf{u}}(\mathbf{u})\,\mathbf{f}_{\mathrm{X}}(\mathbf{u}), \tag{4.32}$$



$\frac{\partial}{\partial u}\mathcal{P}_{\mathrm{B,D}}$
$\mathcal{P}_{\mathrm{B,D}}$
$\frac{\partial}{\partial v}\mathcal{P}_{\mathrm{B,D}}$

Figure 4.3: Tangential plane for the displacement of the viewing ray. The tangential plane is built by the Jacobian matrix of $\mathcal{P}_{\mathrm{B,D}}$. This is a tangent to the unit sphere as $\left\|\mathcal{P}_{\mathrm{B,D}}\right\| = 1$ and the basis $(\frac{\partial}{\partial u}\mathcal{P}_{\mathrm{B,D}}, \frac{\partial}{\partial v}\mathcal{P}_{\mathrm{B,D}})$ is curvilinear.

where the function $\mathbf{f}_{\mathrm{X}} : \mathbb{R}^2 \to \mathbb{R}^2$ is the displacement function. This equation can be used for any direction model where the derivative of $\mathcal{P}_{\mathrm{B,D}}$ is smooth. We will show how the tangent space is computed in the case of the generic B-spline distortion camera model (Eq. (4.31)).

Using the derivative of the sinc function

$$\frac{\partial\,\mathrm{si}(r)}{\partial r} = \frac{\cos(r) - \mathrm{si}(r)}{r} \tag{4.33}$$

and the derivative of the norm function

$$\frac{\partial}{\partial \mathbf{u}}\|\mathbf{f}_{\mathrm{D}}(\mathbf{u})\| = \frac{1}{\|\mathbf{f}_{\mathrm{D}}(\mathbf{u})\|}\left(\frac{\partial \mathbf{f}_{\mathrm{D}}}{\partial \mathbf{u}}(\mathbf{u})\right)^{\top}\mathbf{f}_{\mathrm{D}}(\mathbf{u}), \tag{4.34}$$

the basis vectors are

$$\frac{\partial \mathcal{P}_{\mathrm{B,D}}}{\partial \mathbf{u}} = \begin{pmatrix} \mathrm{si}(r)\frac{\partial \mathbf{f}_{\mathrm{D}}}{\partial \mathbf{u}} + \frac{\cos(r)-\mathrm{si}(r)}{r^2}\left(\frac{\partial \mathbf{f}_{\mathrm{D}}}{\partial \mathbf{u}}\right)^{\top}\mathbf{f}_{\mathrm{D}}\,\mathbf{f}_{\mathrm{D}}^{\top} \\ -\,\mathrm{si}(r)\mathbf{f}_{\mathrm{D}}^{\top}\left(\frac{\partial \mathbf{f}_{\mathrm{D}}}{\partial \mathbf{u}}\right) \end{pmatrix}. \tag{4.35}$$

We dropped the function arguments to increase readability.

This function is continuous if the derivative of $\mathbf{f}_\mathrm{D}$ is continuous. At $r = \|\mathbf{f}_\mathrm{D}\| = 0$, the limit exists, and the basis vectors are

$$\frac{\partial \mathcal{P}_{\mathrm{B,D}}}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial \mathbf{f}_\mathrm{D}}{\partial \mathbf{u}} \\ \mathbf{0}^\top \end{pmatrix}. \tag{4.36}$$

For $\mathbf{f}_\mathrm{X}$, we use a uniform B-spline. We choose the prior in such a way that the camera model behaves locally like a central model. This means neighboring viewing rays have the same displacement. We achieved this by using the smoothness of the second spline derivative (see Section 3.2.7).

4.4 Jacobian Matrix of Implicit Forward Camera Models

Of all proposed non-central camera models, only the backward camera model can be formulated as an analytic expression. If the forward projection is needed for camera calibration, we can invert the backward camera model numerically: Given a world point $\mathbf{p} \in \mathbb{R}^3$, we search for the corresponding image point $\mathbf{u} \in \mathbb{R}^2$. This can be done by solving an optimization problem where the norm of the distance vector $\mathbf{r}(\mathbf{u})$ between the viewing ray and the world point is minimized:

$$\mathbf{r}(\mathbf{u}) = \big(\mathbf{p} - \mathcal{P}_{\mathrm{B,X}}(\mathbf{u})\big) \times \mathcal{P}_{\mathrm{B,D}}(\mathbf{u}). \tag{4.37}$$

As an iterative solver is used, an initial value of the pixel position is needed. Since the world point $\mathbf{p}$ is generated by a measurement which is usually in image space, we use the measured image position for initialization. This leads to an optimization problem which is well defined and can be solved in only a few steps.

But as we use the forward projection model during optimization, the Jacobian matrix is needed as well. Trying to use numeric differentiation can fail because for its calculation, we need to select a step size. If the step size is too small, numerical issues will falsify the result, and if the step size is too large, the approximation of the derivative worsens. As an optimization algorithm is used for inverting the backward camera model, the termination criterion needs to be adapted to the step size of the finite difference. Propagating the step size to the backward camera model is often not possible. The only option is to use a fixed termination criterion. This leads to inaccurate Jacobian matrices. In general, using numeric differentiation leads to slower convergence and is very expensive in terms of computation time.

We therefore propose another approach which directly calculates the derivative of the forward camera model using the backward model. The derivation is closely related to the implicit function theorem and the inverse function theorem. We start with a proposition which is later used to derive the Jacobian matrix of an implicit forward camera model.

Let $\mathbf{f}\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ and $\mathbf{g}\colon \mathbb{R}^n \to \mathbb{R}^m$ be continuous differentiable functions where $\mathbb{R}^n \times \mathbb{R}^m$ have coordinates $(\mathbf{x}, \mathbf{y})$, such that $\mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x})) = \mathbf{0}$, $\forall \mathbf{x} \in \mathbb{R}^n$. We take the point $(\mathbf{x}^*, \mathbf{y}^*)$ which satisfies $\mathbf{g}(\mathbf{x}^*) = \mathbf{y}^*$ and thus $\mathbf{f}(\mathbf{x}^*, \mathbf{y}^*) = \mathbf{0}$ and calculate the derivative $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ at $(\mathbf{x}^*, \mathbf{y}^*)$ by means of $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ as

$$
\begin{aligned}
& & \mathbf{0} &= \mathbf{f}(\mathbf{x}^*, \mathbf{g}(\mathbf{x}^*)) \\
\Rightarrow & & \mathbf{0} &= \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}))}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} \\
\Leftrightarrow & & \mathbf{0} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^*, \mathbf{y}^*) + \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{x}^*, \mathbf{y}^*)\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\mathbf{x}^*) \\
\Leftrightarrow & & \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{x}^*, \mathbf{y}^*)\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\mathbf{x}^*) &= -\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^*, \mathbf{y}^*) \\
\Rightarrow & & \frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\mathbf{x}^*) &= -\left[\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{x}^*, \mathbf{y}^*)\right]^+ \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}^*, \mathbf{y}^*), \qquad (4.38)
\end{aligned}
$$

where $[\cdot]^+$ denotes the Moore-Penrose inverse. We can think of $\mathbf{f}$ being the inverse function of $\mathbf{g}$ in an implicit form.

We will apply this relation to derive the derivative of the forward camera model. We restate the definition of the forward and backward camera model functions

$$\mathcal{P}_\mathrm{F}(\mathbf{p}, \mathbf{k}_\mathrm{I}) = \mathbf{u}, \tag{4.39}$$

$$\mathcal{P}_\mathrm{B}(\mathbf{u}, \mathbf{k}_\mathrm{I}) = \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{d} \end{pmatrix}, \tag{4.40}$$

$$\mathbf{p} = \mathbf{x}_0 + s\mathbf{d}, \tag{4.41}$$

where $\mathbf{p}$ is a 3D point, $\mathbf{k}_\mathrm{I}$ are the intrinsic parameters, $\mathbf{u}$ is an image point, $\mathbf{x}_0$ is the viewing ray base point, $\mathbf{d}$ is the viewing ray direction and $s \in \mathbb{R}$ is the distance between the base point and the world point. In contrast to the former definition of the camera model, we now treat the camera model function not only as a function of $\mathbf{u}$ or $\mathbf{p}$ but also of the intrinsic camera parameters $\mathbf{k}_\mathrm{I}$. The goal is to calculate $\frac{\partial \mathcal{P}_\mathrm{F}}{\partial(\mathbf{p},\mathbf{k}_\mathrm{I})}$ using $\mathcal{P}_\mathrm{B}$.

To calculate the derivative at $(\mathbf{p}^*, \mathbf{k}_\mathrm{I}^*)$, we use the point $(\mathbf{p}^*, \mathbf{k}_\mathrm{I}^*, \mathbf{u}^*)$, where $\mathbf{u}^*$ is determined by numerically inverting $\mathcal{P}_\mathrm{B}$. Plugging Eqs. (4.40) and (4.41) together we get

$$\mathbf{p}^* = \mathcal{P}_{\mathrm{B,X}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*) + s\,\mathcal{P}_{\mathrm{B,D}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*). \tag{4.42}$$

At the fix point, the distance s between the base point and the world point can be determined by projecting the vector pointing from $\mathbf{x}_0$ to $\mathbf{p}^*$ onto $\mathbf{d}$, which results in

$$s = \left(\mathbf{p}^* - \mathcal{P}_{\mathrm{B,X}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*)\right) \cdot \mathcal{P}_{\mathrm{B,D}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*), \tag{4.43}$$

assuming $\left\|\mathcal{P}_{\mathrm{B,D}}(\mathbf{u}, \mathbf{k}_\mathrm{I})\right\| = 1,\ \forall \mathbf{u}, \mathbf{k}_\mathrm{I}$.

Combining Eq. (4.42) with Eq. (4.43), we get the implicit function equation

$$\mathbf{0} = \underbrace{\mathcal{P}_{\mathrm{B,X}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*) + \left[\left(\mathbf{p}^* - \mathcal{P}_{\mathrm{B,X}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*)\right) \cdot \mathcal{P}_{\mathrm{B,D}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*)\right] \mathcal{P}_{\mathrm{B,D}}(\mathbf{u}^*, \mathbf{k}_\mathrm{I}^*) - \mathbf{p}^*}_{\mathbf{f}(\mathbf{p}^*, \mathbf{k}_\mathrm{I}^*, \mathbf{u}^*)}. \tag{4.44}$$

As $\mathbf{u}^* = \mathcal{P}_\mathrm{F}(\mathbf{p}^*, \mathbf{k}_\mathrm{I}^*)$ we can apply Eq. (4.38) to calculate the derivatives $\frac{\partial \mathcal{P}_\mathrm{F}}{\partial \mathbf{p}}(\mathbf{p}^*, \mathbf{k}_\mathrm{I}^*)$ and $\frac{\partial \mathcal{P}_\mathrm{F}}{\partial \mathbf{k}_\mathrm{I}}(\mathbf{p}^*, \mathbf{k}_\mathrm{I}^*)$. In the following, we shall drop the star and write $(\mathbf{p}, \mathbf{k}_\mathrm{I}, \mathbf{u})$ instead of $(\mathbf{p}^*, \mathbf{k}_\mathrm{I}^*, \mathbf{u}^*)$ for the sake of readability.

We start with the derivative with respect to the world point $\mathbf{p}$. We apply Eq. (4.38) and get

$$\frac{\partial \mathcal{P}_\mathrm{F}}{\partial \mathbf{p}}(\mathbf{p}, \mathbf{k}_\mathrm{I}) = -\left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{p}, \mathbf{k}_\mathrm{I}, \mathbf{u})\right]^+ \frac{\partial \mathbf{f}}{\partial \mathbf{p}}(\mathbf{p}, \mathbf{k}_\mathrm{I}, \mathbf{u}). \tag{4.45}$$

The derivatives are

$$\begin{aligned}\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = &\left(\mathbf{I}_3 - \mathcal{P}_{\mathrm{B,D}}\,\mathcal{P}_{\mathrm{B,D}}^{\top}\right)\frac{\partial \mathcal{P}_{\mathrm{B,X}}}{\partial \mathbf{u}} + \\ &\left(\mathcal{P}_{\mathrm{B,D}}^{\top}\left(\mathbf{p} - \mathcal{P}_{\mathrm{B,X}}\right)\mathbf{I}_3 + \mathcal{P}_{\mathrm{B,D}}\left(\mathbf{p} - \mathcal{P}_{\mathrm{B,X}}\right)^{\top}\right)\frac{\partial \mathcal{P}_{\mathrm{B,D}}}{\partial \mathbf{u}}\end{aligned} \tag{4.46}$$

and

$$\frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \mathcal{P}_{\mathrm{B,D}}\,\mathcal{P}_{\mathrm{B,D}}^{\top} - \mathbf{I}_3. \tag{4.47}$$

We denote $\mathbf{I}_3$ as the 3×3 identity matrix. The function arguments were dropped to increase readability.

Similarly applying Eq. (4.38) to calculate the derivative of the forward camera model with respect to the intrinsic parameters $\mathbf{k}_\mathrm{I}$ yields

$$\frac{\partial \mathcal{P}_\mathrm{F}}{\partial \mathbf{k}_\mathrm{I}}(\mathbf{p}, \mathbf{k}_\mathrm{I}) = -\left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{p}, \mathbf{k}_\mathrm{I}, \mathbf{u})\right]^{+}\frac{\partial \mathbf{f}}{\partial \mathbf{k}_\mathrm{I}}(\mathbf{p}, \mathbf{k}_\mathrm{I}, \mathbf{u}). \tag{4.48}$$

The derivative $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{p}, \mathbf{k}_\mathrm{I}, \mathbf{u})$ is shown in Eq. (4.46) and $\frac{\partial \mathbf{f}}{\partial \mathbf{k}_\mathrm{I}}(\mathbf{p}, \mathbf{k}_\mathrm{I}, \mathbf{u})$ is:

$$\begin{aligned}\frac{\partial \mathbf{f}}{\partial \mathbf{k}_\mathrm{I}} = &\left(\mathbf{I}_3 - \mathcal{P}_{\mathrm{B,D}}\,\mathcal{P}_{\mathrm{B,D}}^{\top}\right)\frac{\partial \mathcal{P}_{\mathrm{B,X}}}{\partial \mathbf{k}_\mathrm{I}} + \\ &\left(\mathcal{P}_{\mathrm{B,D}}^{\top}\left(\mathbf{p} - \mathcal{P}_{\mathrm{B,X}}\right)\mathbf{I}_3 + \mathcal{P}_{\mathrm{B,D}}\left(\mathbf{p} - \mathcal{P}_{\mathrm{B,X}}\right)^{\top}\right)\frac{\partial \mathcal{P}_{\mathrm{B,D}}}{\partial \mathbf{k}_\mathrm{I}}.\end{aligned} \tag{4.49}$$

To summarize, Eqs. (4.45) to (4.49) are used to calculate the derivative of the forward camera model by using the backward camera model. Compared to numeric differentiation, these formulas are much more efficient to compute and numerically more stable.

4.5 Influence of Windshields on Camera Models

For autonomous driving, cameras are often placed behind windshields. This has many advantages like protecting the camera against the environment like rain or snow. During rain, wipers remove the rain drops which would otherwise result in distorted images. The drawback is that the windshield itself introduces

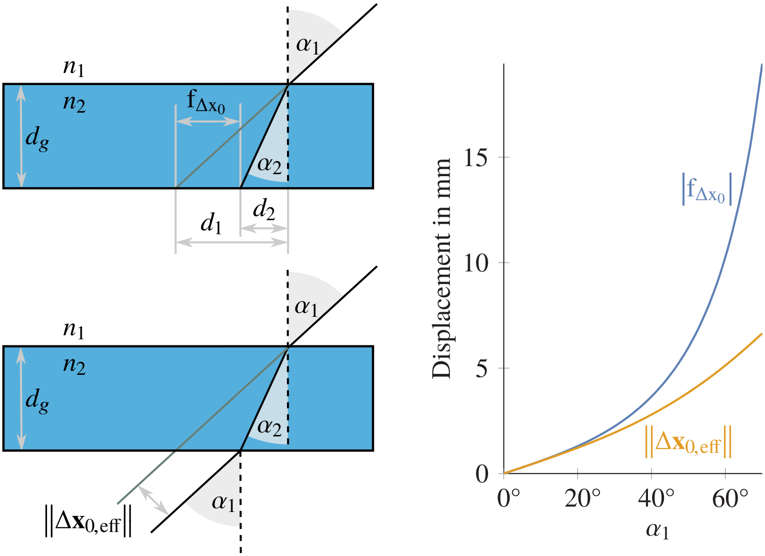


Figure 4.4: Distortion of a ray by a glass pane. We would like to determine two values, the displacement of the ray on the outgoing surface $f_{\Delta x_0}$ and the displacement of a ray traveling through the glass $\Delta\mathbf{x}_{0,\mathrm{eff}}$. On the right, the displacement in both cases is plotted for $n_1 = 1$, $n_2 = 1.5$ and $d_g = 10\,\mathrm{mm}$.

distortions of the viewing rays. In this section, the introduced distortions are investigated and analyzed in detail.

We shall start with Snell's law which describes the refraction of a ray when traveling from one medium to another. In our case the media involved would be air and glass. Snell's law is defined as

$$n_1 \sin(\alpha_1) = n_2 \sin(\alpha_2), \tag{4.50}$$

where n_1 and n_2 are the refractive indices of the first and second medium, and α_1 and α_2 are the angles measured from the surface normals to the incoming and outgoing ray respectively (see Fig. 4.4).

We are interested in two cases: How much is a ray distorted a) at the back surface of the glass, for instance in the case of a protection glass pane in front of a monitor used for active pattern calibration (see Section 5.1), and b) when traveling through a glass pane. In the first case, we are interested in the displacement $\mathrm{f}_{\Delta \mathrm{x}_0}$ of the ray relative to the normal of the glass pane (see upper left Fig. 4.4). In the second case, we have two interfaces, the first one where the ray enters the glass and the second one where the ray exits the glass. Since the enter and exit media are the same, the direction of the ray is not changed, and only the base point is shifted. Therefore we are interested in the shift of the base point (see lower Fig. 4.4). We shall start with a derivation of the first case followed by the second.

In order to derive the geometric relationships, we use th top schematic in Fig. 4.4. The relationships between the angles α_1 and α_2 and the displacements d_1 and d_2 are

$$\tan(\alpha_i) = \frac{d_i}{d_g}, i \in \{1, 2\}, \tag{4.51}$$

where d_g denotes the thickness of the glass, d_1 the position at which the ray would hit the bottom surface without a glass and d_2 with a glass. In this manner, we can derive the displacement $\mathrm{f}_{\Delta \mathrm{x}_0} : \mathbb{R} \to \mathbb{R}$ depending on the incident angle α_1:

$$\begin{aligned} \mathrm{f}_{\Delta \mathrm{x}_0}(\alpha_1) &= d_2 - d_1 && (4.52) \\ &= d_g(\tan(\alpha_2) - \tan(\alpha_1)). && (4.53) \end{aligned}$$

Now we determine α_2 by using Snell's law (Eq. (4.50)):

$$\begin{aligned} \alpha_2 &= \arcsin(n_r \sin(\alpha_1)), && (4.54) \\ n_r &= \frac{n_1}{n_2}. && (4.55) \end{aligned}$$

Combining this with Eq. (4.53) results in

$$\begin{aligned} \mathrm{f}_{\Delta \mathrm{x}_0}(\alpha_1) &= d_g(\tan(\arcsin(n_r \sin(\alpha_1))) - \tan(\alpha_1)) && (4.56) \\ &= d_g\left(\frac{n_r \sin(\alpha_1)}{\sqrt{1 - n_r^2 \sin^2(\alpha_1)}} - \tan(\alpha_1)\right). && (4.57) \end{aligned}$$

This function is plotted on the right-hand side of Fig. 4.4. The last step is now to derive the vector-valued displacement given a line direction $\mathbf{d}$ where $\|\mathbf{d}\| = 1$. We assume that the direction vector is specified in a coordinate system in which the z axis is normal to the pane surface. In our case n_r is smaller than or equal 1, as, e.g., the refractive index of air is ≈ 1 and that of glass lies between 1.4 and 1.9. Using these assumptions we get

$$\Delta\mathbf{x}_0 = \mathrm{f}_{\Delta\mathrm{x}_0}(\arccos(|d_z|))\frac{1}{\sqrt{1-d_z^2}}\begin{pmatrix}0\\0\\d_z\end{pmatrix}. \tag{4.58}$$

Finally, the displaced point is

$$\mathbf{x}_{0,\mathrm{D}} = \mathbf{x}_0 + \Delta\mathbf{x}_0. \tag{4.59}$$

In the second case where a ray passes through a glass pane, we have two transitions: one from air to glass and a second from glass to air. This case is very similar to the first, but now we are not only interested in the displaced point but also want to know the resulting ray. As the ray direction is the same after transitioning through a medium with parallel surface normals, we can directly use Eq. (4.59) for the base point. This results in the distorted line

$$\mathbf{x}_{0,\mathrm{D}} = \mathbf{x}_0 + \Delta\mathbf{x}_0, \tag{4.60}$$

$$\mathbf{d}_\mathrm{D} = \mathbf{d}. \tag{4.61}$$

To measure the effective displacement $\Delta\mathbf{x}_{0,\mathrm{eff}}$, only the amount perpendicular to the ray direction is relevant. We compute $\Delta\mathbf{x}_{0,\mathrm{eff}}$ by projecting the displacement vector onto the plane whose normal vector coincides with the ray direction:

$$\Delta\mathbf{x}_{0,\mathrm{eff}} = \left(\mathbf{I} - \mathbf{d}\mathbf{d}^\top\right)\Delta\mathbf{x}_0. \tag{4.62}$$

The norm of the effective displacement depending on the incident angle is shown on the right-hand side of Fig. 4.4.

5 Camera Calibration

Camera calibration is the process of estimating the parameters of a camera model as well as the transformation between cameras. The parameters of a camera model are called the intrinsic parameters and the transformation between cameras extrinsic parameters.

The calibration process can be divided into two parts, the recording of input data and the estimation of parameters. Several approaches with varying levels of complexity of both steps are proposed. Obviously, one would strive for an easy setup, a simple process for recording the calibration input data and a fully automatic estimation process which ideally yields a precise result. However, all of these goals cannot be achieved simultaneously.

In this work, we focus on a calibration process that uses targets with special markers. This helps to make the process both more robust and precise. As the markers are specially designed, outliers can be detected and rejected more easily, and the position of the markers can be determined with an accuracy up to subpixel scale. The drawback of using special targets is that it requires them to be built and arranged around the calibration object, which can be time-consuming and cumbersome.

Since a single image is not sufficient to calibrate the intrinsic and extrinsic parameters accurately, we record a sequence of images where the target is viewed under various angles. To estimate the intrinsic and extrinsic parameters, we need to transform the markers from the target frame to the camera frame. These transformations must be either estimated jointly with the camera parameters or else they must be measured by an external sensor. Measuring them increases the complexity of the calibration setup significantly as an additional sensor is needed. This sensor then needs to be synchronized with the camera in order to be capable of accurately measuring the transformations.

For each camera, a camera model needs to be selected (see Chapter 4).

There are specific camera models which can be used for only a limited number of different lenses and more general models which can be used for a wide range of lenses. More specific camera models, like global camera models, have the benefit of using a lower number of intrinsic parameters whereas more general ones, like local camera models, have hundreds or even thousands of parameters for each camera. To select a more specific camera model, good knowledge of the target platform is needed. On the other hand, more general camera models increase the risk of overfitting, and generally more data is needed for calibration, which leads to a longer calibration process. One of the most general camera models is a discrete camera model where each pixel is mapped to a different viewing ray. Such a model can also describe non-continuous camera models as the viewing rays of each pixel are independent of each other. In order to calibrate such a model, we need at least two distinct measurements per pixel, which usually imposes either some interpolation between sparse measured markers or requires a more complex target which provides dense measurements.

In the following chapters, two different approaches to camera calibration are introduced: a calibration process that uses a display providing dense measurements as target (Section 5.1) and a calibration process with checkerboard targets which makes for a simpler setup (Section 5.2). The active display camera calibration process can be used to estimate discrete camera models whereas the checkerboard camera calibration is suitable for local and global camera models.

5.1 Active Display Camera Calibration

In this section, we present a camera calibration process using a display. The display allows to generate measurements for every camera pixel in which the display is visible.

In our case, the display is a computer monitor. A camera is mounted on a three-axis linear positioning system that points towards a monitor. The camera is moved using the linear positioning system. For each position, various patterns are displayed and recorded by the camera. The goal is to get multiple 3D point measurements for each camera pixel in a global coordinate system. These measurements are then used to estimate the extrinsic and intrinsic parameters of the camera model. The coordinate systems used in the following chapters are the camera coordinate system $\{C\}$, which is fixed at a camera, the rig

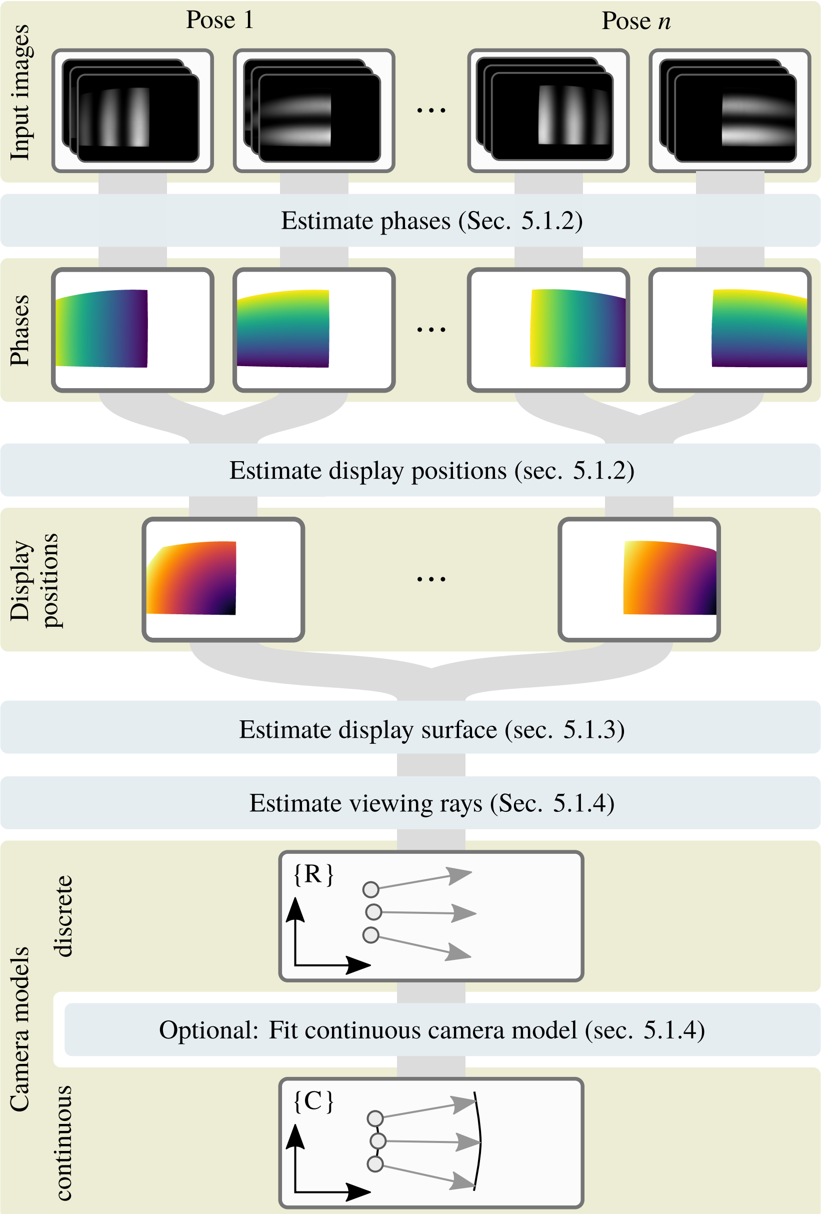

Pose 1
Pose *n*
Input images
...
Estimate phases (Sec. 5.1.2)
Phases
...
Estimate display positions (sec. 5.1.2)
Display positions
...
Estimate display surface (sec. 5.1.3)
Estimate viewing rays (Sec. 5.1.4)
Camera models
discrete
{R}
Optional: Fit continuous camera model (sec. 5.1.4)
continuous
{C}

Figure 5.1: Process overview of the active display camera calibration.

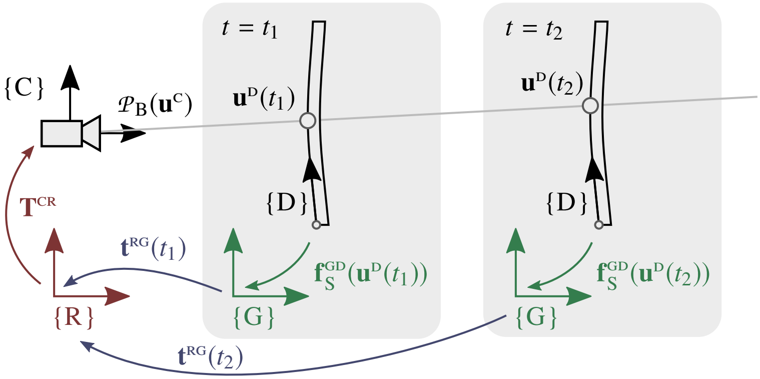


Figure 5.2: Overview of the coordinate systems and the variables used for transforming the display point measurement $\mathbf{u}^D$ into the camera coordinate system $\{C\}$ for a single camera. For this transformation we need the display surface function $\mathbf{f}_S^{GD}$, the measured time dependent translation $\mathbf{t}^{RG}(t)$ and the extrinsic camera pose $\mathbf{T}^{CR}$.

coordinate system $\{R\}$, the global coordinate system $\{G\}$, which is fixed in the world, and the display coordinate system $\{D\}$ (see Fig. 5.2).

The overview of the calibration process is shown in Fig. 5.1. Since we are using a display for calibration, the display itself needs to be calibrated (Section 5.1.1). Secondly, we introduce a method where the 2D position $\mathbf{u}^D$ on the display can be estimated for each camera pixel in which it is visible (Section 5.1.2). For calibration, the display point needs to be mapped to a 3D point in rig coordinates $\mathbf{p}^R$ using the mapping function $\mathbf{f}_S^{GD}$ (Section 5.1.3). The last step is to estimate a camera model based on these 3D points and the extrinsic pose $\mathbf{T}^{CR}$ (Section 5.1.4). In the following sections, all of these steps are explained in detail.

5.1.1 Display Calibration

Since a display is used for calibration, the display itself needs to be calibrated. This section is about gray value calibration and not about geometric calibration (see Section 5.1.3 for display surface estimation). To calibrate the display, a camera is placed in front of it, and a constant gray value is displayed. We record one image for every discrete monitor gray value. Example images for different

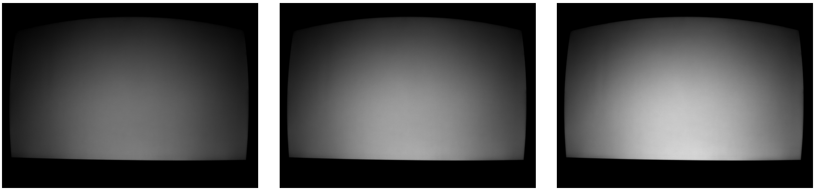


Figure 5.3: Recorded gray images during display calibration.

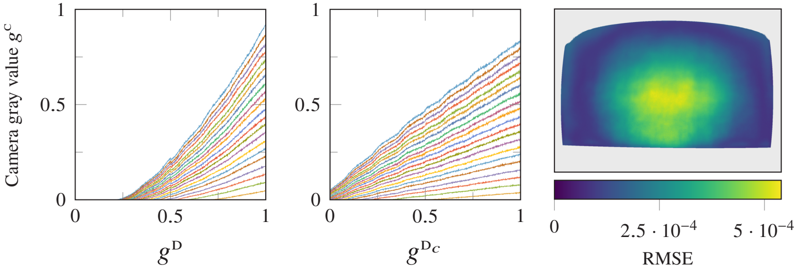


Figure 5.4: Display calibration. The response curve of a display is shown before calibration (on the left) and after calibration (in the middle). On the right, the RMSEs for a normalized gray value range of each monitor pixel is shown.

gray values are shown in Fig. 5.3. It is clearly visible that the intensity depends on the viewing angle. The camera gray value decreases for higher viewing angles. On the left-hand side of Fig. 5.4, the measured gray values are shown as a function of the displayed gray value at different image locations. We see that these curves are non-linear as computer monitors usually apply gamma correction. For the later calibration steps, we need a linear curve. To get a linear relationship between g^{C} and g^{D}, the displayed gray is corrected by

$$g^{\mathrm{D}c} = (g^{\mathrm{D}})^{\frac{1}{\gamma}}, \tag{5.1}$$

where g^{D} is the input intensity and γ the gamma value used to correct the image. The gamma value is usually between 1.8 and 2.2. In addition, small display gray values are clipped by the camera. This means that the intensity of the lowest gray value needs to be increased.

To correct the non-linearity, the camera pixel with the highest amplitude is chosen, and the residual

$$e_r = \begin{cases} \mathrm{f_g}(g_0^{\mathrm{D}}) - g^{\mathrm{C}}, & g^{\mathrm{D}} < g_0^{\mathrm{D}} \\ \mathrm{f_g}(g^{\mathrm{D}}) - g^{\mathrm{C}}, & \text{otherwise} \end{cases}, \tag{5.2}$$

$$\mathrm{f_g}(g^{\mathrm{D}}) = a(g^{\mathrm{D}} - g_0^{\mathrm{D}})^{\exp(\gamma_e)} + g_0^{\mathrm{C}}, \tag{5.3}$$

is used in a NLS problem. The optimization parameters are the gray value shift g_0^{D}, which is used to account for the display gray values below the black level the camera, the gamma value γ_e mapped to a positive value using the exponential function to prevent negative power values, the amplification a, and the camera gray value offset g_0^{C}.

To get to a linear curve between the display value and the measured gray value, the fitted function $\mathrm{f_g}$ is inverted for $g^{\mathrm{D}} > g_0^{\mathrm{D}}$, which leads to

$$\mathrm{f_g^{-1}}(g^{\mathrm{C}}) = \left(\frac{g^{\mathrm{C}} - g_0^{\mathrm{C}}}{a}\right)^{\frac{1}{\exp(\gamma_e)}}. \tag{5.4}$$

This function is applied to every display pixel, assuming that the damping of the display light is not position-dependent and independent of the viewing direction.

To test our approach, we again record an image for each discrete gray value but apply $\mathrm{f_g^{-1}}$ to each value. The resulting response curves and the RMSEs for a normalized gray value range are plotted in Fig. 5.4. We see that all response curves are almost linear and thus the assumption of the independence of the viewing angle holds true.

5.1.2 Dense Measurement Generation

The process of how dense measurements can be generated using a display is mainly adapted from [Rap12]. Sinusoidal fringe patterns are displayed where the phase encodes the position on the display. These patterns have the advantage of creating highly accurate position estimates even if the display is not in the focus plane. Placing the display in the focus plane is not always possible as multiple depths need to be recorded and the estimation accuracy increases for

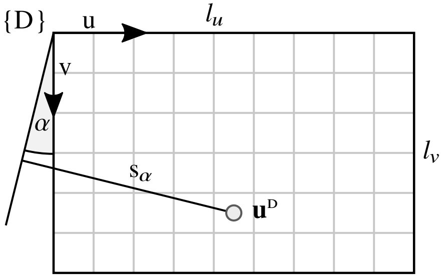


Figure 5.5: Visualization of the geometric relations to calculate the gray value for each display pixel position $\mathrm{g}^{\mathrm{D}}_{i,k,\alpha}(\mathbf{u}^{\mathrm{D}})$. The display size in pixels is denoted as (l_u, l_v).

close display positions. Another advantage is that the spatial discretization of the camera pixels does not alter the estimation of the phase shift.

The sinusoidal fringe pattern, which will be shown on the display, is described as (see also Fig. 5.5)

$$\mathrm{g}^{\mathrm{D}}_{i,k,\alpha}(\mathbf{u}^{\mathrm{D}}) = \frac{1}{2} + \frac{1}{2}\cos\left[2\pi\left(k\frac{\mathrm{s}_\alpha(\mathbf{u}^{\mathrm{D}})}{l} + \frac{i}{N}\right)\right], \tag{5.5}$$

$$\mathrm{s}_\alpha(\mathbf{u}^{\mathrm{D}}) = \begin{pmatrix}\cos(\alpha)\\ \sin(\alpha)\end{pmatrix} \cdot \mathbf{u}^{\mathrm{D}}, \tag{5.6}$$

$$l = \sqrt{l_u^2 + l_v^2}, \tag{5.7}$$

where $\mathrm{g}^{\mathrm{D}}_{i,k,\alpha}$ is the display pixel value, α is the wave angle, k is the wave number, $i \in \{0, \ldots, N-1\}$ the phase shift index, N the number of phase shifts and l_u and l_v are the number of pixels in the horizontal and vertical axis of the display. Some example patterns are shown in Fig. 5.6. As monitors have a non-linear response curve, this gray value is transformed with $\mathrm{f}_{\mathrm{g}}^{-1}$ (see Section 5.1.1).

Given a set of recorded camera images $\mathrm{g}^{\mathrm{C}}_i$ for each k and α, where i is the phase shift index shown on the display, the goal is to estimate a cosine wave for each pixel. A cosine wave can be described by a phase w_ϕ, offset w_{c} and amplitude w_{a} for each pixel:

$$\mathrm{g}^{\mathrm{C}}_i(\mathbf{u}^{\mathrm{C}}) = \mathrm{w}_{\mathrm{c}}(\mathbf{u}^{\mathrm{C}}) + \mathrm{w}_{\mathrm{a}}(\mathbf{u}^{\mathrm{C}})\cos\left(\mathrm{w}_\phi(\mathbf{u}^{\mathrm{C}}) + 2\pi\frac{i}{N}\right). \tag{5.8}$$

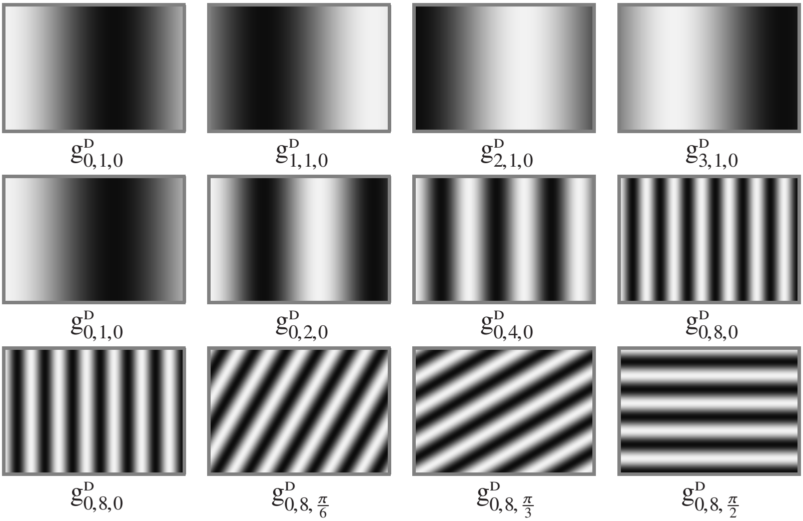


Figure 5.6: Sinusoidal fringe patterns which are shown on a display for dense measurement generation. Shown are the normalized gray values of $g^{D}_{i,k,\alpha}(\mathbf{u}^{D})$ (see Eq. (5.5)) for different phase shift indices i (first row), different wave numbers k (second row) and different wave angles α (last row).

These parameters can be estimated in closed form using a linear least squares formulation (see [Rap12])

$$w_{a}(\mathbf{u}^{C}) = \frac{2}{N}\left|\sum_{i=0}^{N-1} g_{i}^{C}(\mathbf{u}^{C}) \exp\left(-2\pi \mathrm{i}\frac{i}{N}\right)\right|, \tag{5.9}$$

$$w_{\phi}(\mathbf{u}^{C}) = \arg\left[\sum_{i=0}^{N-1} g_{i}^{C}(\mathbf{u}^{C}) \exp\left(-2\pi \mathrm{i}\frac{i}{N}\right)\right], \tag{5.10}$$

$$w_{c}(\mathbf{u}^{C}) = \frac{1}{N}\sum_{i=0}^{N-1} g_{i}^{C}(\mathbf{u}^{C}), \tag{5.11}$$

where i is the imaginary unit of a complex number.

By comparing Eq. (5.5) with Eq. (5.8), one can determine a relationship between the estimated phase and the display position s_α:

$$\frac{l}{2\pi k}\mathrm{w}_\phi(\mathbf{u}^{\mathrm{C}}) \overset{!}{=} \mathrm{s}_\alpha(\mathbf{u}^{\mathrm{D}}). \tag{5.12}$$

To estimate $\mathbf{u}^{\mathrm{D}}$ for each camera pixel $\mathbf{u}^{\mathrm{C}}$, multiple wave angles are needed. Given multiple wave angles and using Eq. (5.6), a linear least squares formulation is used to estimate the display position $\mathbf{u}^{\mathrm{D}}$:

$$\begin{pmatrix} \mathrm{w}_{\phi,\alpha_1}(\mathbf{u}^{\mathrm{C}}) \\ \mathrm{w}_{\phi,\alpha_2}(\mathbf{u}^{\mathrm{C}}) \\ \vdots \end{pmatrix} = \frac{2\pi k}{l} \begin{pmatrix} \cos(\alpha_1) & \sin(\alpha_1) \\ \cos(\alpha_2) & \sin(\alpha_2) \\ \vdots \end{pmatrix} \mathbf{u}^{\mathrm{D}}. \tag{5.13}$$

To increase the accuracy, a high wave number is desirable, but this will make the phase reconstruction ambiguous. To overcome this problem, a multi-phase shift approach combined with a Gray code similar to [Rap12] is used.

To detect whether a measurement is reliable and if the camera pixel is illuminated by the display, the estimated amplitude $\mathrm{w}_{\mathrm{a}}(\mathbf{u}^{\mathrm{C}})$ is used. This amplitude can be used for an uncertainty measure as lower amplitudes yield higher uncertainties (see [Rap12]). Therefore, a threshold is used to remove non-reliable measurements.

So far we have described the process of measuring the display positions $\mathbf{u}^{\mathrm{D}}$ at the corresponding image positions $\mathbf{u}^{\mathrm{C}}$. To estimate the camera model, we need to transform the display positions to the rig frame for which a mapping from the display surface to the fixed world coordinate system is needed. The modeling and estimation of this mapping is explained in the next section.

5.1.3 Display Surface Estimation

In this section, the estimation of the mapping between the 2D display coordinate system $\{\mathrm{D}\}$ and a fixed world coordinate system $\{\mathrm{G}\}$ is described (see Fig. 5.2). We will specify that mapping using two different assumptions: a flat display versus a non-flat display.

When assuming a flat display, the mapping can be divided into two transformations: one between {D} and {G}, denoted as $\mathbf{T}^{\mathrm{GD}}$, and the other between the display pixel coordinates $\mathbf{u}^{\mathrm{D}}$ and the 3D coordinates $\mathbf{p}^{\mathrm{D}}$. In general, $\mathbf{T}^{\mathrm{GD}}$ contains a translational and a rotational part. However in our setup, the camera is only translated. This means, that the translation $\mathbf{t}^{\mathrm{GD}}$ cannot be distinguished from the displacement of the viewing rays in the rig coordinate system. So we can only estimate the rotational part and set the translational part to zero. In order to estimate the transformation between the display pixel $\mathbf{u}^{\mathrm{D}}$ and the metric display coordinate $\mathbf{p}^{\mathrm{D}}$, the size of a monitor pixel in meter needs to be known. Since we assume a flat display, the transformation function $\mathbf{f}^{\mathrm{GD}}_{\mathrm{S,F}}\colon \mathbb{R}^2 \to \mathbb{R}^3$, $\mathbf{u}^{\mathrm{D}} \mapsto \mathbf{p}^{\mathrm{G}}$, is

$$\mathbf{f}^{\mathrm{GD}}_{\mathrm{S,F}}(\mathbf{u}^{\mathrm{D}}) = \mathbf{R}^{\mathrm{GD}} \begin{pmatrix} u^{\mathrm{D}}_u s_u \\ u^{\mathrm{D}}_v s_v \\ 0 \end{pmatrix}, \tag{5.14}$$

where $\mathbf{s} = (s_u, s_v)^\top$ is the size of a monitor pixel in u- and v-direction in meters and $\mathbf{R}^{\mathrm{GD}}$ is the rotation matrix. In total, there are five parameters to be estimated.

Unfortunately, real displays are not perfectly flat. To model a non-flat display, we use a uniform B-spline function $\mathbf{f}^{\mathrm{GD}}_{\mathrm{S,C}}\colon \mathbb{R}^2 \to \mathbb{R}^3$ (see Section 3.1) and transform $\mathbf{u}^{\mathrm{D}}$ by

$$\mathbf{p}^{\mathrm{G}} = \mathbf{f}^{\mathrm{GD}}_{\mathrm{S,C}}(\mathbf{u}^{\mathrm{D}}). \tag{5.15}$$

The parameters to estimate are the control points of the spline.

The next question is how to estimate the parameters of $\mathbf{f}^{\mathrm{GD}}_{\mathrm{S}}$. As neither the intrinsic nor the extrinsic camera parameters are known, these parameters are jointly estimated with $\mathbf{f}^{\mathrm{GD}}_{\mathrm{S}}$. The measurements are the display points $\mathbf{u}^{\mathrm{D}}$, and we assume that all measurements of the same camera pixel transformed into the camera coordinate system must be located on the same line. Given a set of display point measurements $\mathbf{u}^{\mathrm{D}}_{\mathbf{a}}(t_i)$ for one camera pixel $\mathbf{a}$, the points are first transformed into the camera coordinate system by

$$\mathbf{p}^{\mathrm{R}}_{\mathbf{a}}(t_i) = \mathbf{t}^{\mathrm{RG}}(t_i) + \mathbf{f}^{\mathrm{GD}}_{\mathrm{S}}(\mathbf{u}^{\mathrm{D}}_{\mathbf{a}}(t_i)), \tag{5.16}$$

where $\mathbf{t}^{\mathrm{RG}}(t_i)$ is the measured rig translation at time t_i. Then the distance of the line expressed as $\mathbf{x}^{\mathrm{R}}_{0,\mathbf{a}}, \mathbf{d}^{\mathrm{R}}_{\mathbf{a}}$ at the camera pixel $\mathbf{a}$ is used to calculate the display

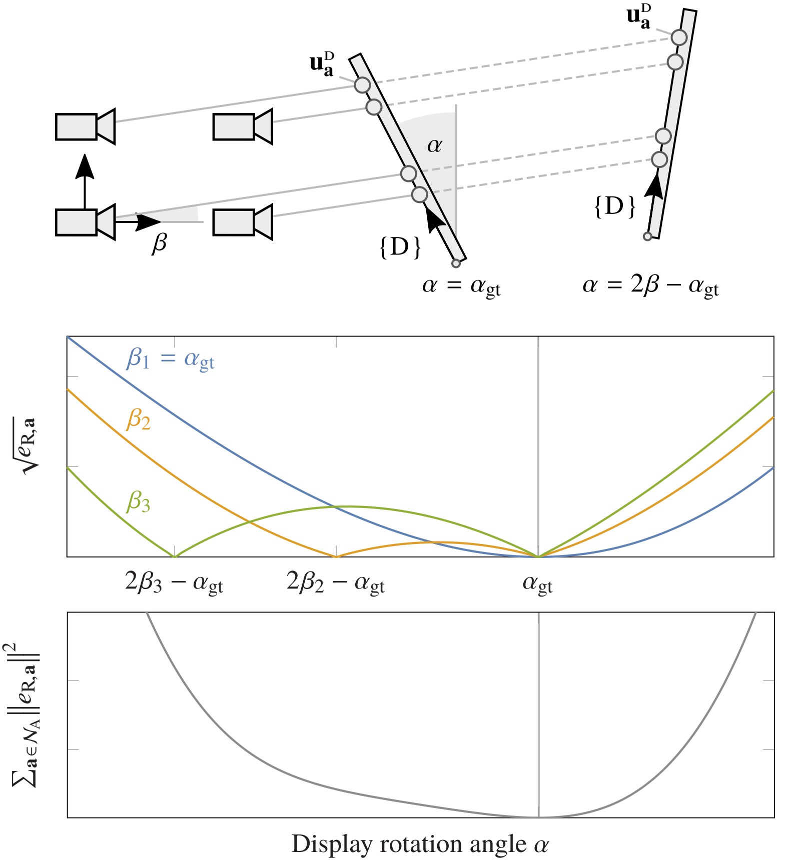


Figure 5.7: Visualization of the residual used for estimating the display surface mapping function in a 2D simulation. The top image shows the simulation setup. In the middle, the display point measurement distance to the optimal viewing ray $\sqrt{e_{\mathrm{R},\mathbf{a}}}$ is plotted over the display angle α for different viewing rays β_i. For $\beta \neq \alpha_{\mathrm{gt}}$, there is a second minimum at $2\beta - \alpha_{\mathrm{gt}}$. The bottom plot shows the final residual with a single minimum at α_{gt}.

surface parameters $\mathbf{k}_\mathrm{S}$. This results in a nested optimization problem, which can be expressed as

$$\underset{\mathbf{k}_\mathrm{S}}{\arg\min} \sum_{\mathbf{a}\in\mathcal{N}_\mathrm{A}} \left\| e_{\mathrm{R},\mathbf{a}} \right\|^2, \tag{5.17}$$

$$e_{\mathrm{R},\mathbf{a}} = \min \sum_{i=1}^{N_{\mathrm{t},\mathbf{a}}} \left\| \left(\mathbf{p}^{\mathrm{R}}_{\mathbf{a}}(t_i) - \mathbf{x}^{\mathrm{R}}_{0,\mathbf{a}} \right) \times \mathbf{d}^{\mathrm{R}}_{\mathbf{a}} \right\|^2, \tag{5.18}$$

where $\mathcal{N}_\mathrm{A}$ is the set of all camera pixels and $N_{\mathrm{t},\mathbf{a}}$ is the number of display measurements at the camera pixel $\mathbf{a}$. The minimization problem in Eq. (5.18) can be solved by using a SVD (see Section 3.2.5).

To further investigate the properties of this residual, a 2D simulation with a flat display surface is used (see Fig. 5.7). The translation vector $\mathbf{t}^{\mathrm{RG}}$ and the viewing rays $\mathbf{x}^{\mathrm{R}}_0, \mathbf{d}^{\mathrm{R}}$ are all in the xy-plane. The values of the second dimension of $\mathbf{u}^{\mathrm{D}}$ are all zero, and the rotation can be expressed by a single angle α, which leads to the display surface function

$$\mathbf{f}^{\mathrm{GD}}_{\mathrm{S},2\mathrm{D}}(\mathbf{u}^{\mathrm{D}}) = \begin{pmatrix} \cos(\alpha) & 0 \\ \sin(\alpha) & 0 \\ 0 & 0 \end{pmatrix} \mathbf{u}^{\mathrm{D}}. \tag{5.19}$$

To generate the necessary measurements, we assign a viewing ray with the base point $\mathbf{x}_{0,\mathbf{a}} = \mathbf{0},\ \forall\mathbf{a}$ and the direction $\mathbf{d} = (\cos(a_1), \sin(a_1), 0)^\top, \mathbf{a} = (\beta, 0)^\top$ to each camera pixel. We set $\alpha = \alpha_{\mathrm{gt}}$ and create different $\mathbf{u}^{\mathrm{D}}_{\mathbf{a}}(t_i)$ by varying $\mathbf{t}^{\mathrm{RG}}(t_i)$.

Finally, since we have only a single parameter to estimate, we plot the residual $\sqrt{e_{\mathrm{R},\mathbf{a}}}$ over α for three different angles β_i, $i \in \{1, 2, 3\}$ with $\beta_1 = \alpha_{\mathrm{gt}}$ and $\beta_1 < \beta_2 < \beta_3$ (see middle plot in Fig. 5.7). It can be seen that the residual is zero at $\alpha = \alpha_{\mathrm{gt}}$. But if $\beta \neq \alpha_{\mathrm{gt}}$, we get a second minimum at $2\beta - \alpha_{\mathrm{gt}}$. These two cases are shown in the top image of Fig. 5.7. In the bottom plot, the final residual is shown. Due to the variation of the second minimum, there is a single minimum for the final residual. So in order to get a residual with one global minimum, multiple viewing rays and the display surface function need to be optimized jointly.

As the residuum is a non-linear function, a NLS solver is used. This formulation of the residuum has two downsides which are both related to the best line fit in the residual calculation where the inner problem (Eq. (5.18)) is solved using a SVD as mentioned in Section 3.2.5. The first downside is that any NLS solver needs the derivative of the residuum function. Considering that close to the optimum a SVD of an almost rank-deficient matrix is needed, an analytical derivative is non-trivial and numerical differentiation is unstable (see [PL00]). The second downside is the single residual per viewing. This makes each residual dependent on a lot of different display positions, and the problem gets dense even if a compact function like a uniform B-spline is used for $\mathbf{f}_{\mathrm{S}}^{\mathrm{GD}}$.

To overcome these downsides, the problem is reformulated by moving the best line fit estimation to the outer NLS problem. This means that in addition to the display surface function parameters, the line parameters for each viewing ray are estimated. The optimization problem can be defined as

$$\operatorname*{arg\,min}_{\mathbf{k}_{\mathrm{S}},\mathbf{X}_0^{\mathrm{R}},\mathbf{D}^{\mathrm{R}}} \sum_{i=1}^{N_t} \sum_{\mathbf{a}\in\mathcal{N}_{\mathbf{a},i}} \left\| \left(\mathbf{p}_{\mathbf{a}}^{\mathrm{R}}(t_i) - \mathbf{x}_{0,\mathbf{a}}^{\mathrm{R}} \right) \times \mathbf{d}_{\mathbf{a}}^{\mathrm{R}} \right\|^2, \tag{5.20}$$

where $\mathbf{k}_{\mathrm{S}}$ are the parameters of the display surface $\mathbf{f}_{\mathrm{S}}^{\mathrm{GD}}$ and $\mathbf{X}_0^{\mathrm{R}}$, $\mathbf{D}_0^{\mathrm{R}}$ are the viewing ray parameters for every camera pixel. $\mathcal{N}_{\mathrm{A},i}$ denote the set of all camera pixels containing a measurement at the time point t_i. As the viewing rays are over-parametrized, we use the local line parametrization from Section 3.2.4.

This problem formulation is sparse since each residual can now be split up into each display point at the expense of having more parameters to estimate. Also, the local support of the uniform B-spline function of the non-flat display surface function contributes to the sparsity.

A visualization of the non-flat surface can be found in the evaluation chapter in Fig. 7.13.

After having estimated the display surface, all measurements can be transformed into the rig frame. The final step of the active display calibration process is the estimation of the camera model using these measurements. This will be explained in the next section.

5.1.4 Camera Model Estimation

The final step of the active display calibration process is the generation of a camera model using the estimated display points in the rig frame. Depending on further applications, either a discrete or a continuous camera model is chosen. The estimation of both choices will be investigated in the following two sections.

Discrete Camera Model

The discrete camera model consists of one ray per pixel (see Section 4.2). Such a camera model is very flexible as it can cope with non-continuous and non-central lenses.

To estimate the rays, a coordinate system needs to be defined. The simplest case is to omit the camera coordinate system and estimate the rays in the rig coordinate system. This can be done by transforming the display point measurements to the rig coordinate system and estimate one ray per pixel by minimizing the squared distance between the ray and the 3D points (see Section 3.2.5). The extrinsic camera poses are implicitly captured by the relation between the viewing rays.

If a camera coordinate system is desired, the coordinate transformation between the rig frame and the camera frame needs to be defined. The translational part could be defined as the closest point to all viewing rays for each camera. This point can be determined by solving a LLS problem (see Section 3.2.6). For the rotational part of the transformation, there is no obvious choice. One way to define this part is to use the center pixel in the image aligned with the z axis and the image u axis aligned with the x axis. The y axis must be perpendicular to the x and y axes to form an orthogonal basis. Another way is to choose a global camera model which best models the used lens and to estimate its parameters by fitting the global camera model to the discrete camera model (see next section). This has the benefit of transferring the characteristics of the camera coordinate system of a global camera model, like having an optical center which is aligned with the z axis, to the discrete model. This can be important for further algorithms such as image rectification or undistortion, as sometimes these algorithms rely on such properties.

Continuous Camera Model

Continuous camera models can be mainly divided into local and global types. Non-central local camera models like the B-spline distortion camera model (see Section 4.3) are preferred as they are flexible enough to accurately describe the discrete camera model. The high number of parameters that local camera models require can be robustly estimated due to the availability of dense measurements.

Since camera model functions are non-linear, a NLS problem is formulated and solved. As a residual, the distance between the viewing rays either to the display points or to the estimated discrete camera model in the rig frame can be used. The parameters to estimate are the extrinsic and intrinsic camera model parameters. We use the local B-spline distortion camera model (see Section 4.3) wherever a continuous model is needed. Due to the non-linear problem formulation, an initialization is needed.

For initialization, the lens projection model and rough estimates of the focal length and the principal point need to be known beforehand. These parameters can usually be derived from the used camera and the mounted lens. If an identity transformation for the extrinsic transformation is used to estimate the non-central distortion B-spline camera model, the NLS solver converges really slowly. This is why a three-stage approach is used. In the first stage, the distortion-free central global camera model (see Section 4.1.1) with fixed intrinsic parameters is fitted. This provides a rough initialization of the extrinsic camera parameters. In the second stage, a central global generalized camera model with distortions is used. In the final step, the distortion B-spline camera model with fixed extrinsic parameters is fitted. The first two stages are fast to compute due to their low number of parameters. In the first stage, there is only one transformation with six DOF to estimate. In the second stage, eight additional intrinsic camera parameters are determined. The last stage takes longest as the non-central distortion B-spline camera model usually has hundreds of parameters. But due to the close-to-optimal initialization of the previous steps, only few iterations are needed for convergence.

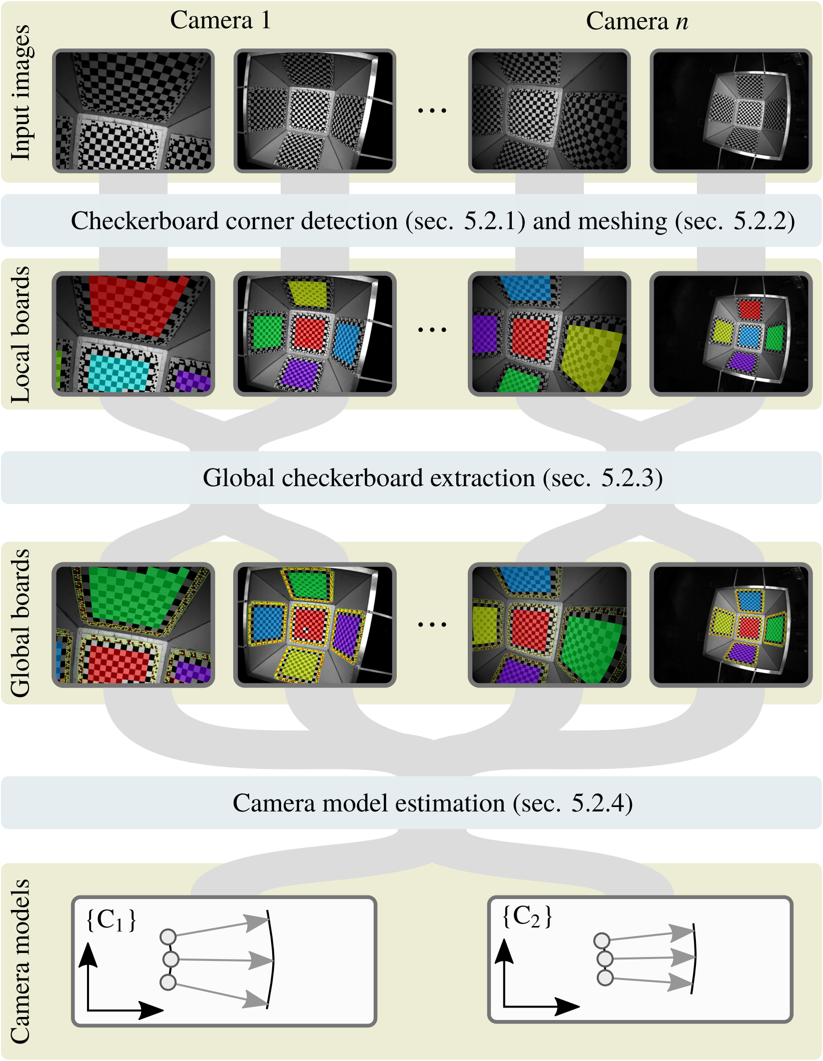

Camera 1
Camera n
Input images
...
Checkerboard corner detection (sec. 5.2.1) and meshing (sec. 5.2.2)
Local boards
...
Global checkerboard extraction (sec. 5.2.3)
Global boards
...
Camera model estimation (sec. 5.2.4)
Camera models
{C$_1$}
{C$_2$}

Figure 5.8: Process overview of the checkerboard camera calibration.

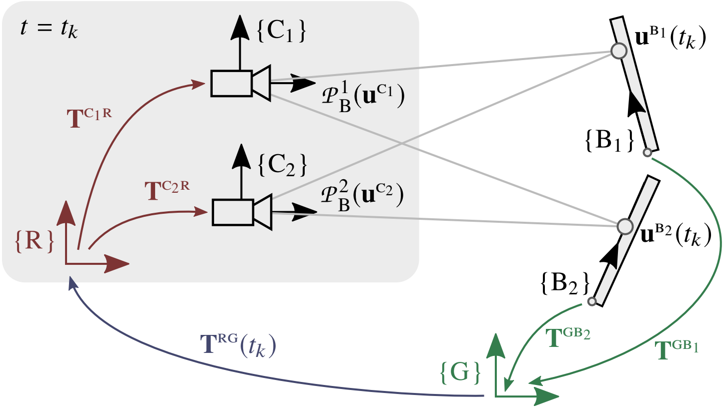


Figure 5.9: Overview of the coordinate system and variables used for transforming the checkerboard position $\mathbf{u}^{B_i}$ to the camera coordinate system $\{C_i\}$.

5.2 Checkerboard Camera Calibration

In this section, we introduce the checkerboard camera calibration. The goal of this calibration process is to make the recording and the camera model estimation simple and fast but still accurate. This is achieved by using independent 2D checkerboards. Such checkerboard are easy to fabricate as they can be printed by generally available printers. The image recording is done by freely moving a camera rig around without the need to measure any pose. A robust and accurate corner detection together with a joint estimation of the checkerboard poses provides an accurate estimate of the intrinsic and extrinsic parameters of the camera model. An overview of the calibration process is visualized in Fig. 5.8.

A lot of different patterns are used in literature, such as checker patterns [Zha00], circular patterns [MH03], checker boards [LM02], concentric circles [JQ05] or April tags [RSO13] to only name a few. A comparison of the detection quality of circular and checkerboard patterns is presented in [MW07]. We use checkerboards as targets and the checkerboard corner positions as measurements as they are easy to build and have one of the smallest detection biases under distortion or when viewed from a flat angle.

The coordinate systems used in the next sections of this work are the global coordinate system {G}, the checkerboard coordinate system {B}, the rig coordinate system {R} and the camera coordinate system {C} (see Fig. 5.9). It is assumed that both the transformation between the camera and the rig and the transformation between the global and the checkerboard coordinate system are time-invariant during calibration. Only the transformation between the global and the rig coordinate system changes over time.

The checkerboard camera calibration follows the work presented in [8] and [Str15]: The checkerboard extraction process is described in Sections 5.2.1 to 5.2.3, followed by the final camera model estimation which estimates the intrinsic and extrinsic camera model parameters given the checkerboard corners in Section 5.2.4. We give only a brief overview. A more detailed description can be found in [Str15].

5.2.1 Checkerboard Corner Detection

Checkerboard corner detection is split into two parts: The first step is a classification problem to find checkerboard corner candidates, followed secondly by corner refinement to achieve subpixel accuracy.

A sufficiently small region around an ideal checkerboard corner has two properties which we use for detection: The region is point-symmetric, and the image gradient vectors align with the tangent to a circle centered at the checkerboard corner. These two features are calculated for the whole image. The norm of the feature vector directly correlates with how similar the recorded corner is to an ideal corner (the smaller the values, the closer). Therefore, we classify all pixels as checkerboard corners where each component of the feature vector is smaller than that of an empirically selected threshold. In addition, we only label pixels as corners if their image gradient orientation feature has a local minimum.

As a next step, these corner candidates are then refined to achieve subpixel accuracy. This is done as in [GMCS12]. The image gradient of a neighbor pixel should be perpendicular to the connection vector between the corner candidate and this neighbor pixel. This leads to a LLS problem which can be solved efficiently.

To increase the overall corner detection performance, we filter out corners in saturated image regions or in low contrast regions as they tend to be less accurate. To detect corners of different checkerboard sizes, an image pyramid is used. To speed up the detection, the corner classification is implemented on a GPU using CUDA.

5.2.2 Checkerboard Corner Meshing

The goal of corner meshing is to group all corners of a single board together and then to assign the same local corner ID to each of these corners. The meshing process is divided into three parts: pairing corners, building checkerboard patches and creating checkerboards.

Two corners form a pair if they are adjacent along a checkerboard edge. Corners with no neighbors are filtered out. The corner pairs are then used to form board patches. A board patch consist of four pairs with each corner always being part of two pairs. To create a checkerboard, a starting patch is selected randomly, and a local board coordinate system is attached. Then neighboring patches, meaning patches that share one corner pair, are added to this checkerboard until no new neighboring patch is found. Afterwards, a new checkerboard is started by randomly selected a patch from the remaining ones. This is repeated until no more patches are available.

5.2.3 Global Checkerboard Extraction

Given checkerboard detections with local corner indices, the goal is now to assign a checkerboard ID to each corner which is the same for all measurements on the same checkerboard. Also, we need the 2D global corner index, which is the global corner position with respect to a fixed coordinate system attached to that checkerboard.

We achieve this by adding a binary code around the checkerboard (see Fig. 5.10). This code is made up of hexagons, which reduces the chance of falsely detecting checkerboard corners on the binary code strips as hexagons corners are not point-symmetric. Nine hexagons at a time form a code patch which is as big as one checkerboard tile. The bit pattern of each code patch is random. After checkerboard extraction, the code around the board is read by extrapolation. As

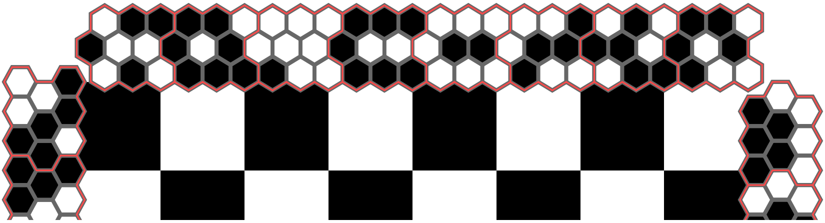


Figure 5.10: The checkerboard code around the board. Each hexagon encodes one bit of information. The hexagons are then distributed around each checkerboard. Nine hexagons at a time form a code patch (marked in red). All bit patterns of all patches together with their position and the global board ID are stored in a codebook. The read code patch is looked up in the codebook and used to extract the global board ID as well as the global checkerboard corner indices.

a good camera model is needed for extrapolation, we calibrate each camera with the local boards and local corner indices. The read code patch is then looked up in a codebook. The codebook stores all known code patches together with their corresponding positions and global checkerboard IDs. We use multiple code patches and a voting scheme to determine the best match in the codebook as there are only $2^9 = 512$ different codes which is not sufficient for many different or large boards. From the best match, the coordinate mapping is calculated to transform the local corner indices to the global ones. Also, the global board ID is assigned to each corner of that board.

5.2.4 Camera Model Estimation

Given the global checkerboard corners, the whole calibration problem is formulated as a NLS problem by minimizing the reprojection error. The reprojection error $\mathbf{e}_{\mathrm{R,i}}$ is defined (see also Fig. 5.9) as

$$\mathbf{e}_{\mathrm{R,i}}(t_k) = \mathbf{u}_i^{\mathrm{C}_l}(t_k) - \mathcal{P}_{\mathrm{F}}^{l}\left(\mathbf{T}^{\mathrm{C}_l\mathrm{R}}\mathbf{T}^{\mathrm{RG}}(t_k)\mathbf{T}^{\mathrm{GB}_j}\mathbf{p}_i^{\mathrm{B}_j}(t_k)\right), \tag{5.21}$$

$$\mathbf{p}_i^{\mathrm{B}_j}(t_k) = \begin{pmatrix} \mathbf{u}_i^{\mathrm{B}_j}(t_k) \\ 0 \end{pmatrix}. \tag{5.22}$$

The checkerboard corner $\mathbf{p}^{\mathrm{B}_j}$ is first transformed into the camera coordinate system and then projected onto the image using the camera forward model

$\mathcal{P}_{\mathrm{F}}$. The reprojection error is the vector between the projected point and the measured checkerboard corner position $\mathbf{u}^{C}$ in the image.

As the camera is moved around, we could estimate one checkerboard pose per camera image $\mathbf{T}^{\mathrm{C}_l \mathrm{B}_j}(t_k)$. We reduce the number of poses, especially in multi-camera multi-checkerboard setups, by taking into account that the transformation between the cameras and the transformations between the checkerboards do not change over time. As the camera rig is moved around, we estimate k rig poses $\mathbf{T}^{\mathrm{RG}}(t_k)$. The coordinate origin of $\{\mathrm{G}\}$ and $\{\mathrm{R}\}$ are arbitrary, so we select $\{\mathrm{G}\}$ to be equal to the first checkerboard $\mathbf{T}^{\mathrm{GB}_1} = \mathbf{I}$ and $\{\mathrm{R}\}$ to be equal to the first camera $\mathbf{T}^{\mathrm{C}_1\mathrm{R}} = \mathbf{I}$.

To summarize, the parameters that need to be optimized are the checkerboard poses $\mathbf{T}^{\mathrm{GB}_j}, j \geq 2$, the rig poses $\mathbf{T}^{\mathrm{RG}}(t_k), k \geq 1$, the extrinsic camera poses $\mathbf{T}^{\mathrm{C}_l\mathrm{R}}, l \geq 2$ and the intrinsic camera parameters $\mathbf{k}_{\mathrm{I}}$ of $\mathcal{P}_{\mathrm{F}}^{l}$ of each camera.

These parameters are estimated using a NLS solver with the following problem formulation

$$\underset{\mathbf{k}_{\mathrm{I}},\mathbf{T}^{\mathrm{C}_l\mathrm{R}},\mathbf{T}^{\mathrm{RG}}(t_k),\mathbf{T}^{\mathrm{GB}_j}}{\arg\min} \sum_{k=1}^{N_t} \sum_{i=1}^{N_{i,k}} \left\| \mathbf{e}_{\mathrm{R,i}}(t_k) \right\|^2, \tag{5.23}$$

where N_t is the number of recorded time points and $N_{i,k}$ is the number of detected checkerboard corners at time t_k.

6 Calibration Performance Assessment

Calibration performance assessment answers the question of how good a calibration is. We propose two calibration performance assessment methods, the reprojection error histogram (REH) and the camera model difference (CMDI), which will both be explained in the following two sections.

6.1 Reprojection Error Histogram

The reprojection error is defined as the difference between a 3D point $\mathbf{p}$ projected onto the image plane by using a camera model forward projection function $\mathcal{P}_{\mathrm{F}}$ and a measurement $\hat{\mathbf{u}}$ of the same 3D point in image space. In case of camera calibration with checkerboards, the measurement is a detected corner in the image space and the 3D point is determined by the transformation between the checkerboard corner and the camera. The reprojection error is defined as

$$\mathbf{e}_{\mathrm{R}} = \hat{\mathbf{u}} - \mathcal{P}_{\mathrm{F}}(\mathbf{p}). \tag{6.1}$$

Since we are using multiple checkerboard corners, we also have multiple reprojection errors $\mathbf{e}_{\mathrm{R,i}}$. We assume that the detector noise of the measurements is normally distributed and thus the reprojection errors should as well be normally distributed. In literature, the RMS reprojection error

$$e_{\mathrm{R,RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{e}_{\mathrm{R,i}}\right\|^2}, \tag{6.2}$$

is often used as a performance measure, but this value is only an indicator and the absolute value cannot be used to predict absolute error bounds as we will show later in simulation (see Section 7.4.2). Also, the distribution of the

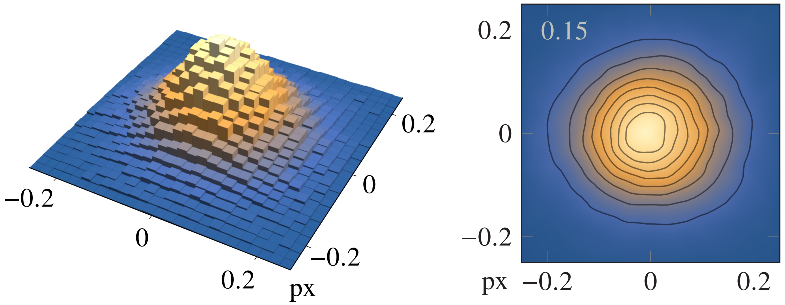


Figure 6.1: Example of a REH. As the reprojection error is 2D, the REH is in 3D. On the left-hand side the 3D REH is shown. On the right-hand side the same histogram is plotted as a contour plot. The height of the histogram is color-coded. The upper left number in the contour plot is the RMS reprojection error.

reprojection error is not visible in this value. So instead of using the RMS reprojection error, we use a 3D histogram of the reprojection error (see Fig. 6.1). Using this kind of histogram has the benefit of making the reprojection error distribution visible. To visualize the histogram we can either plot it in 3D or use a contour plot where the height of the contour plot is color-coded.

6.2 Camera Model Difference

As we would like to compare different calibration processes using different camera models, we seek a descriptive measure of how similar two camera models are. Expressing the similarity by a single number would not be descriptive enough since the location of the difference would be lost. E.g., if the focal lengths of the camera models are slightly different, their difference is zero at the principal point and increases towards the border. So instead of using a single number, we seek a similarity measure function $\mathbf{f}_\mathrm{D}$ which takes two camera models $\mathcal{P}^\mathrm{A}$, $\mathcal{P}^\mathrm{B}$ and an image position $\mathbf{u}$. This can be expressed as

$$\mathbf{e}_\mathrm{S} = \mathbf{f}_\mathrm{D}\left(\mathbf{u}, \mathcal{P}^\mathrm{A}, \mathcal{P}^\mathrm{B}\right). \tag{6.3}$$

The next important question is which unit to use for the per-pixel measure. The required quality of a calibration is usually dependent on the application. E.g., for stereo vision, the images are rectified in such a way that the epipolar lines are horizontal and in the same pixel row. If the matching search direction is only horizontal, it is important that the calibration error in vertical direction is less than one pixel. Another possible application would be the estimation of 3D landmarks in different images. If a feature detector is used in image space and the reprojection error of the landmark is minimized, again, a pixel-based measure would give an idea about the expected error induced by the camera calibration. So we use a per-pixel measure where the error unit is pixel.

The basic algorithm for estimating the difference between two camera models consists of two steps: In the first step, a transformation $\mathbf{T}^{\text{BA}}$ of one camera with respect to the other is determined. Due to possible varying scales of different calibration processes, $\mathbf{T}^{\text{BA}}$ consist of a rotation, a translation and a scale. In the second step, the transformation is fixed and the per-pixel difference is calculated.

For both steps, an optimization problem is solved by projecting 3D points onto each camera image and then minimizing the distance between both of them. Let $\mathbf{f}_\text{P} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ be a function which takes a pixel position and translates it into a 3D point. The residual for the optimization problem can then be expressed as

$$\mathbf{e}_\text{S} = \mathcal{P}^A_\text{F}(\mathbf{f}_\text{P}(\mathbf{u})) - \mathcal{P}^B_\text{F}(\mathbf{T}^{\text{BA}}\mathbf{f}_\text{P}(\mathbf{u})). \tag{6.4}$$

We design $\mathbf{f}_\text{P}$ to be

$$\mathbf{f}_\text{P}(\mathbf{u}) = \overline{\mathcal{P}}^A_{\text{B,X}}(\mathbf{u}) + s\,\mathcal{P}^A_{\text{B,D}}(\mathbf{u})\,. \tag{6.5}$$

We select a reference camera, in this case camera A, and use a point on the viewing ray which belongs to the pixel position for which the residual should be computed. $\overline{\mathcal{P}}^A_{\text{B,X}}(\mathbf{u})$ is the base point on the viewing ray $\mathbf{u}$ which is closest to the camera center and $\mathcal{P}^A_{\text{B,D}}$ is the viewing ray direction. $s \in \mathbb{R} \cup \{\infty\}$ describes the distance between the 3D point and the base point. For central cameras, the camera center is well defined since $\overline{\mathcal{P}}^A_{\text{B,X}}(\mathbf{u}) = \mathbf{c}, \mathbf{c} \in \mathbb{R}^3, \forall \mathbf{u}$. For non-central cameras, we choose the camera center to be the point which is closest to all viewing rays using the algorithm described in Section 3.2.6.

The downside of using a reference camera is that in this case the distance measure is non-symmetric, as the result is different if we swap $\mathcal{P}^\text{A}$ and $\mathcal{P}^\text{B}$. This can be circumvented by calculating the error difference twice, first with camera

A and secondly with camera B as the reference camera, and then calculating the mean between both results. In practice, the error difference turns out to be almost the same for both cases. The upside of using only one camera as reference is that only the non-reference camera needs to be continuous and the reference camera can be either discrete or continuous. This property is used to compare discrete camera models with continuous camera models without fitting a continuous model to the discrete one.

By projecting a 3D point into the image space with $\mathcal{P}^A_\mathrm{B}$, this will lead to the original image position $\mathbf{u}$. Plugging Eqs. (6.4) and (6.5) together results in

$$\mathbf{e}_\mathrm{S}(\mathbf{u}^\mathrm{A}, s) = \mathbf{u}^\mathrm{A} - \mathcal{P}^B_\mathrm{F}\left(\mathbf{T}^\mathrm{BA}\left(\overline{\mathcal{P}}^A_{\mathrm{B,X}}(\mathbf{u}^\mathrm{A}) + s\,\mathcal{P}^A_{\mathrm{B,D}}(\mathbf{u}^\mathrm{A})\right)\right). \tag{6.6}$$

The distance parameter s can be chosen in two ways: regular points in 3D which are influenced by the non-central aspect of the camera, and 3D points at infinity. For regular 3D points we directly use Eq. (6.6). In case of points at infinity, we need to change the formulation. As for points at infinity the displacement of a viewing ray vanishes, only its direction needs to be considered. This is done by searching for the one image point whose viewing ray direction is the same as the transformed viewing ray of the other camera model:

$$\mathbf{e}_\mathrm{S}(\mathbf{u}, \infty) = \mathbf{u} - \operatorname*{arg\,min}_{\mathbf{u}^*} \left\|\mathcal{P}^B_{\mathrm{B,D}}(\mathbf{u}^*) - \mathbf{L}^\mathrm{BA}\mathcal{P}^A_{\mathrm{B,D}}(\mathbf{u})\right\|^2. \tag{6.7}$$

To transform the line direction, we need to use only the linear part of the affine transformation which we denote as $\mathbf{L}^\mathrm{BA}$.

With the residual $\mathbf{e}_\mathrm{S}$, the optimization problem for $\mathbf{T}^\mathrm{BA}$ is formulated as

$$\operatorname*{arg\,min}_{\mathbf{T}^\mathrm{BA}} \sum_i \sum_{j=1}^{\lfloor I_u/n \rfloor} \sum_{k=1}^{\lfloor I_v/n \rfloor} \left\|\mathbf{e}_\mathrm{S}((nj, nk)^\top, s_i)\right\|^2, \tag{6.8}$$

where I_u and I_v are the number of columns and rows of the image respectively and n controls the pixel step size. For high-resolution cameras, the number of residuals can become huge, resulting in high computational costs for solving the optimization problem. But since we assume smooth camera models, it is not necessary to use every pixel. In this case, we may simply increase the pixel step size n in order to reduce the number of residuals.

For our applications like autonomous driving, the non-central part of the camera model can often be neglected in further applications as the objects of interest are far away. For this purpose, we use only $s = \infty$. Since this would lead to a nested optimization problem (see Eqs. (6.7) and (6.8)), we approximate the optimization problem by calculating the difference between the direction vectors directly:

$$\underset{\mathbf{L}^{\mathrm{BA}}}{\arg\min} \sum_{j=1}^{\lfloor I_u/n \rfloor} \sum_{k=1}^{\lfloor I_v/n \rfloor} \left\| \mathcal{P}^{B}_{\mathrm{B,D}}((nj, nk)^{\top}) - \mathbf{L}^{\mathrm{BA}} \mathcal{P}^{A}_{\mathrm{B,D}}((nj, nk)^{\top}) \right\|^2. \tag{6.9}$$

When comparing camera models from different calibration processes, the scale of the 3D space may be slightly different. As this error might dominate the CMDI, we will compensate it. Thus, the affine transformation $\mathbf{T}^{\mathrm{BA}}$ consists of a translation, a rotation and a scale matrix. The rotation matrix $\mathbf{R}^{\mathrm{BA}}$ and the scaling vector $\mathbf{s}_{\mathrm{c}} \in \mathbb{R}^3$ form the linear part of the affine transformation $\mathbf{L}^{\mathrm{BA}}$. The linear part combined with the translational part results in the final affine transformation:

$$\mathbf{L}^{\mathrm{BA}} = \mathbf{R}^{\mathrm{BA}} \operatorname{diag}(\mathbf{s}_{\mathrm{c}}), \tag{6.10}$$

$$\mathbf{T}^{\mathrm{BA}} = (\mathbf{L}^{\mathrm{BA}}, \mathbf{t}^{\mathrm{BA}}). \tag{6.11}$$

To visualize the per-pixel CMDI, we plot the norm of $\mathbf{e}_{\mathrm{S}}$ at $\mathbf{u}$ in the reference camera and color-code it. In Fig. 6.1, an example of the CMDI is shown. We use a pinhole camera model for both $\mathcal{P}^{\mathrm{A}}$ and $\mathcal{P}^{\mathrm{B}}$ with the same principal point at the image center but a different focal length. Since the direction of the optical axis is independent of the focal length, the difference at the principal point is zero. The difference increases with the distance to the optical axis. For this synthetic case, the scaling vector is fixed to vector of all ones as otherwise the difference would be zero.

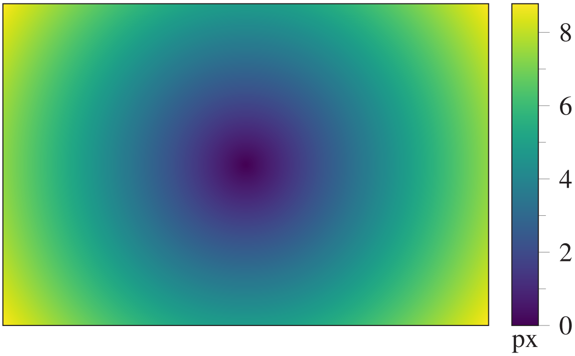


Figure 6.2: The CMDI of two pinhole camera models with slightly different focal lengths. Shown is the color-coded norm projection error $\|\mathbf{e}_S(\mathbf{u}, \infty)\|$ (see Eq. (6.7)). The image size is (1500 px, 1000 px) and the principal point is at (750 px, 500 px). The focal length of the first camera is 1000 px whereas the second focal length is 1010 px.

7 Evaluation

In this chapter we evaluate the performance of the generic global camera model (GCM) and the generic B-spline distortion camera model (BCM) using a checkerboard calibration process as well as the discrete camera model (DCM) using an active display calibration process (see Chapter 4 and Chapter 5). Each camera model is evaluated by comparing it with the ground truth camera model. To measure the difference between two camera models we use the camera model difference (CMDI) introduced in Section 6.2.

We first evaluate the performance of the camera models using the *triple camera setup* consisting of three cameras where one camera can be placed behind a glass pane under various angles. This setup is replicated in simulation and the performance is assessed for all three camera models. In simulation the environment is well-known. This enables us to see the effect of a single parameter on the estimated result, e.g., the pose of the glass pane or the non-planarity of the display. Afterwards, a calibration benchmark is proposed which uses the DCM with an active display calibration process serving as the ground truth camera model. This benchmark is used to assess the quality of the GCM and BCM calibrated with the checkerboard calibration process.

In the last section of this chapter, a multi-camera system for autonomous driving is calibrated to show the performance benefits of the BCM in a real-world scenario.

7.1 Evaluated Camera Models

In this section, we describe the camera models used for evaluation, summarized in Table 7.1. As a baseline, we choose the GCM. This model has a low number of intrinsic parameters, utilizes the widely used lens distortion model and can handle various lens types. This model is compared with the proposed BCM. For the BCM, we use a uniform B-spline of order four and one control point per

Name	Class	Type	# intrinsic parameters	Reference
GCM-C	global	central	8	sec. 4.1
GCM-NC	global	non-central	11	sec. 4.1
BCM-C	local	central	$\lfloor I_u/100 \rfloor \cdot \lfloor I_v/100 \rfloor$	sec. 4.3
BCM-NC	local	non-central	$2 \cdot \lfloor I_u/100 \rfloor \cdot \lfloor I_v/100 \rfloor$	sec. 4.3
DCM	discrete	non-central	$4 \cdot I_u \cdot I_v$	sec. 4.2

Table 7.1: Overview of the camera models used for evaluation. As a baseline, we use the GCM and compare it with the proposed BCM. Both are used in a central and a non-central version. The DCM is used as the ground truth camera model for the calibration benchmark.

100 px. The number of control points was determined empirically. Additionally, a regularization is added to the problem: for the central part the third and for the non-central part the second order derivative smoothness integral (see Section 3.2.7). The parameters of the DCM are estimated using the active display calibration process. This DCM serves as the ground truth camera model of the calibration benchmark.

In the following sections, we refer to camera model names as listed in Table 7.1.

7.2 Triple Camera Setup

The *triple camera setup* consists of three grayscale FLIR Blackfly S cameras (BFS-PGE-50S5M-C) with a resolution of 2448 px × 2048 px. We use a bit depth of 12 bits for recording. All cameras use the same S-mount lens from Lensation with a focal length of 4 mm and an aperture of F/1.8 (Lensagon BM4018S118). Two cameras are mounted on the same horizontal axis and the third one is placed roughly in the middle (see Fig. 7.1). This setup allows to mount a glass pane with a thickness of 10 mm in front of the middle camera. Also, the angle of that pane can be changed. Due to space constraints, the size of the glass pane is limited and, because of the wide FOV for larger angles of the glass pane, only some parts of the incoming light rays travel through it. To

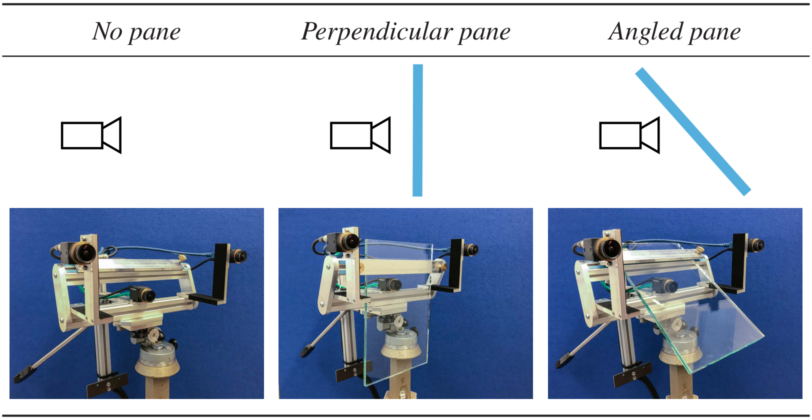


Figure 7.1: The *triple camera setup* consists of three cameras pointing roughly in the same direction. The first setup (left), *no pane*, is without a glass pane, the second one (middle), *perpendicular pane*, with a glass pane perpendicular to the optical axis and the third one (right), *angled pane*, has an angle of about 45° between the optical axis and the glass pane. The glass pane has a thickness of 10 mm.

circumvent that problem, we limit the FOV of the middle camera by setting a region of interest (ROI) resulting in a total resolution of 1528 px × 1100 px.

We use three different settings: The first setting, *no pane*, is without a pane, the second one, *perpendicular pane*, is with a pane perpendicular to the optical axis and the third one, *angled pane*, has a glass pane where the angle between the pane and the optical axis is about 45°.

7.3 Triple Camera Datasets

In this section, the recorded datasets with the *triple camera setup* are described. We start with the dataset for the active display calibration process (Section 7.3.1) and then move on to the checkerboard dataset (Section 7.3.2).

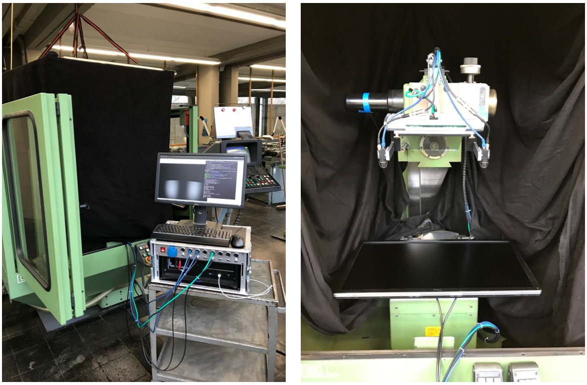


Figure 7.2: The active display calibration setup. We use a milling machine as a linear positioning system and a 4k monitor to display the sinusoidal fringe patterns. A molleton cover is used to reduce the influence of extraneous light.

7.3.1 Active Display Dataset

For the active display calibration dataset, we mount the *triple camera setup* on the linear positioning system, place a display in front of it and record images of different sinusoidal fringe patterns. As linear positioning system, we use a Deckel FP4 milling machine. A Dell UP3216Q monitor with a resolution of 3840 px × 2160 px and a color bit depth of 10 bit serves as display. To reduce the influence of extraneous light, we cover the milling machine with molleton (see Fig. 7.2).

For the sinusodial fringe patterns, we use a wave number of 192, eight face shifts, and eight wave angles equally distributed in the range $[0, \pi/2]$. To resolve the phase reconstruction ambiguity, we use a multi-phase shift approach combined with a Gray code. We use three face shifts and two wave angles and start with a wave number of one. The wave number is doubled until the desired wave number is reached ($2^{n_k}, n_k = 0, \ldots, 7$). Additionally, we As the gray values are directly used to estimate the position on the display, it is important to reduce the noise of the camera image. This is achieved by acquiring the images with the highest possible bit depth of the camera, in this case 12 bit, as well as by taking multiple shots and then averaging the recorded images. Some of the recorded images of a single pose are shown in Fig. 7.3.

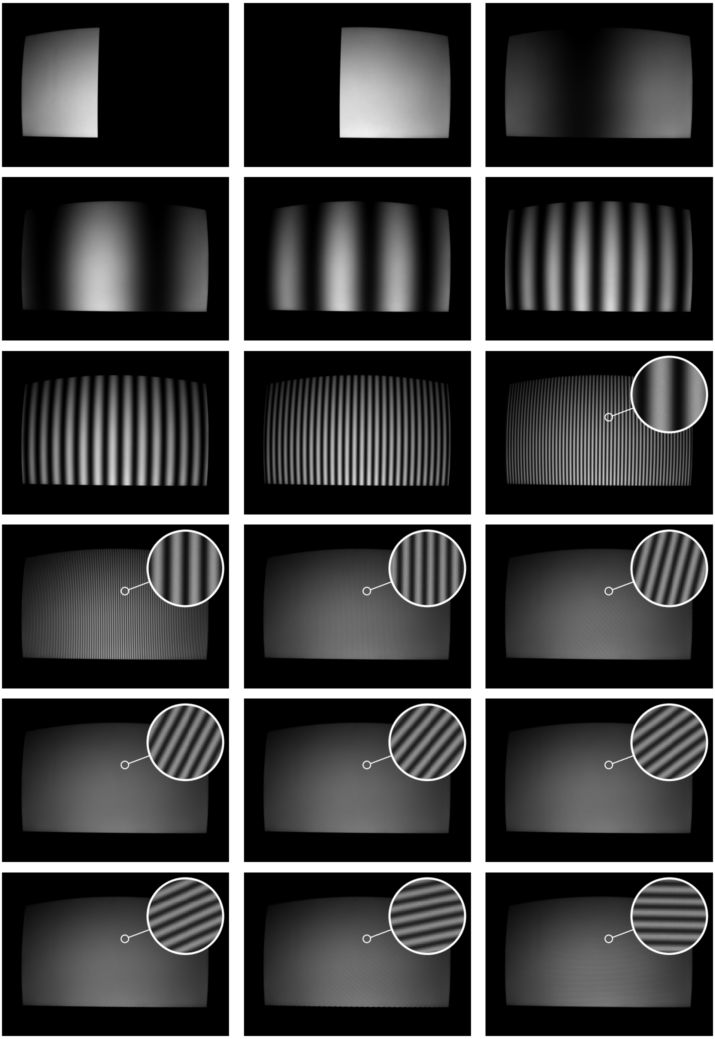


Figure 7.3: Fringe pattern calibration input images of a single display pose. Going from left to right and top to bottom, the first two images show the Gray code, the next eight images show the increase of the wave number (1, 2, 4, 8, 16, 32, 64, 128) used to resolve phase reconstruction ambiguities and the last eight images show the eight different wave angles used at the desired wave number of 192. For higher wave angles, a magnifier with a factor of 10 is added. Best viewed digitally with zoom.

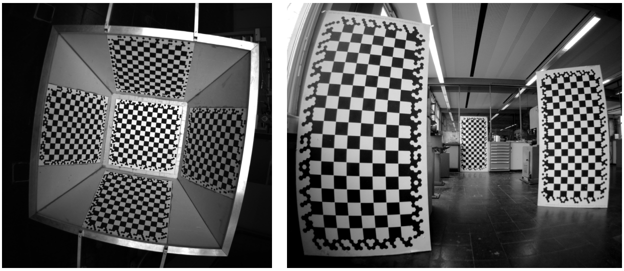


Figure 7.4: The checkerboard targets used for the checkerboard dataset. The truncated pyramid (left) consists of five checkerboards whereas the big boards setting (right) is built by three single checkerboards distributed in a room.

To vary the pose of the display, we use the linear positioning system. The positions are located on an equally spaced $3 \times 3 \times 4$ grid with a cell size of $270\,\mathrm{mm} \times 195\,\mathrm{mm} \times 100\,\mathrm{mm}$.

For each setting, we record 4176 images per camera in total which takes approximately 1.5 h.

7.3.2 Checkerboard Dataset

For the checkerboard dataset, two different checkerboard configurations are used (see Fig. 7.4): on the one hand, five checkerboards arranged in a truncated pyramid and on the other hand, three big boards. The checkerboards of the truncated pyramid consist of 10×10 full tiles with a tile size of $3.6\,\mathrm{cm} \times 3.6\,\mathrm{cm}$. The three big boards, where one board is farther away, consist of 16×6 full tiles with a tile size of $9.6\,\mathrm{cm} \times 9.6\,\mathrm{cm}$. The overall checkerboard size including the binary code is $1\,\mathrm{m} \times 2\,\mathrm{m}$. To perform global association of checkerboards, a binary code is arranged around each board (see Section 5.2.1 for more detail).

To record the image sequences, the camera rig is moved by hand. For all three settings, *no pane*, *perpendicular pane* and *angled pane*, a sequence with the truncated pyramid is recorded. For the two setups with a glass pane, a second sequence with the big boards is recorded to make the estimation of the non-central part of the camera model more robust. To record the sequences, all cameras are synchronized by an external trigger. Also, the exposure time

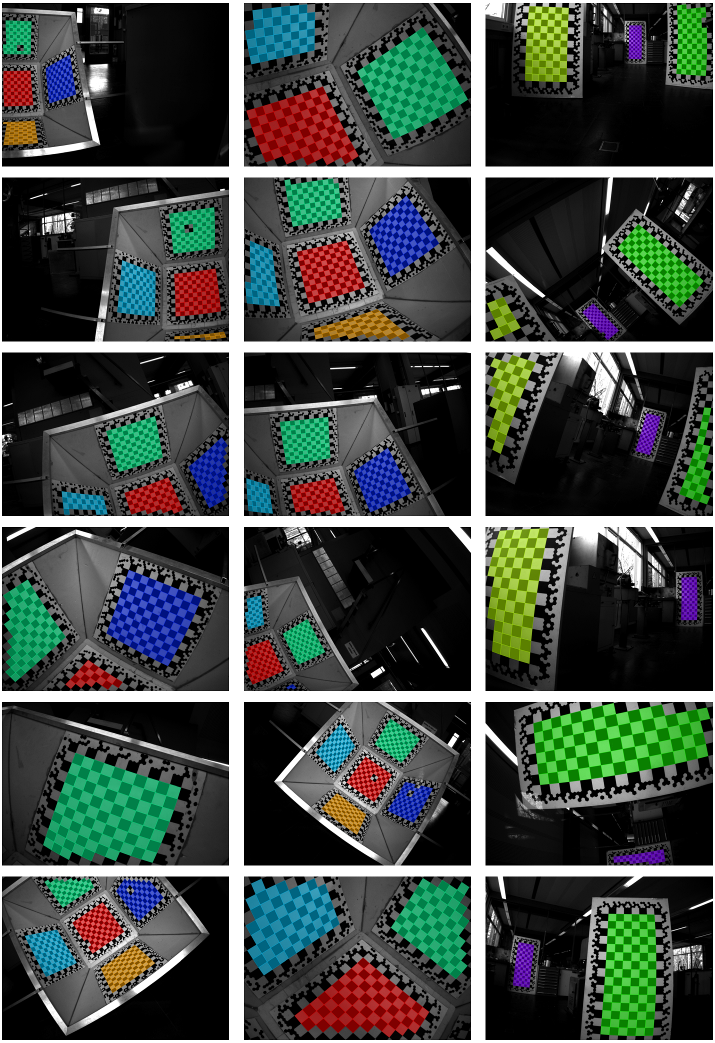


Figure 7.5: For checkerboard calibration, we use two different checkerboard setups. One consists of a truncated pyramid (left two columns) and the other one consists of three 2 m × 1 m checkerboards where one board is farther away (right column).

Setup	# images		# checkerboard corners		
	Truncated pyramid	Big boards	Center camera	Left camera	Right camera
No pane	275	-	93315	136221	136080
Perpendicular pane	325	365	179098	258765	264324
Angled pane	250	282	136554	191759	204449

Table 7.2: Number of recorded images and number of checkerboard corners for each setting of the *triple camera setup* (see Section 7.2).

of all cameras is fixed and set to the same value. The number of images and the number of detected checkerboard corners are listed in Table 7.2 and some recorded images of the dataset and detected corners are shown in Fig. 7.5.

In the following section, we evaluate the estimation accuracy of the camera models (Section 7.1) in simulation using the *triple camera setup* (Section 7.2) by virtually generating the dataset from this section. Afterwards, the real-world dataset is used to create a calibration benchmark (Section 7.5) followed by an experimental evaluation of the GCM and BCM calibrated with the checkerboard calibration process (Section 7.6).

7.4 Simulation

In this section, the active display and checkerboard calibration processes (see Chapter 5) with different camera models (see Section 7.1) are evaluated in simulation. The goal is to assess the performance of these methods in a controlled and noise-free environment.

We create a meaningful simulation by replicating the *no pane* setting of the *triple camera setup* (Section 7.2). We use the same checkerboard targets which are used to record the dataset (Section 7.3.2). The ground truth camera models and the poses of the camera rig are taken from an estimate of a real calibration.

The output of the simulation consists of rendered images which are then used in the calibration process. The rendered images are created by ray casting (see

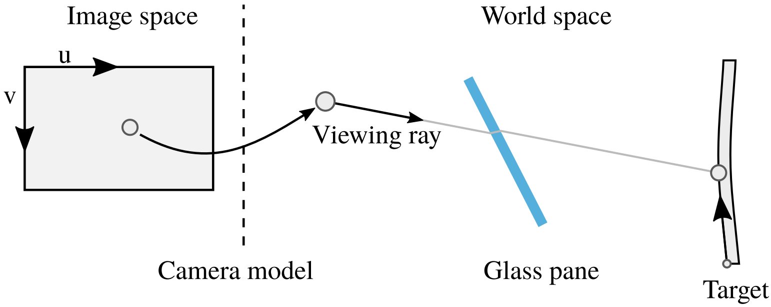


Figure 7.6: The simulation is based on ray casting. For each pixel in image space, the corresponding viewing ray is determined using the camera model. Then the hit point of that ray with the target is calculated and its brightness is determined. If a glass pane is placed between the viewing ray base point and the target, the displacement is calculated by using Snell's law (see Section 4.5).

Fig. 7.6). Starting from a pixel position in image space, we determine the viewing ray using the backward camera model. The viewing ray is transformed into the target coordinate frame and the intersection point with the target is calculated. The target is modeled as a surface and the 2D coordinate on that surface is determined. This 2D coordinate is finally used to determine the brightness of the camera image pixel. To increase simulation accuracy, we use multiple rays per pixel and average the determined brightness values.

In the same simulator, a glass pane can be placed between the camera and the target to create the other two settings, *perpendicular pane* and *angled pane*. We assume a flat pane and calculate the distortion of the ray according to Eqs. (4.60) and (4.61) using the pane normal vector and the ray direction.

The camera lens is modeled only as a geometric mapping between an image pixel and a viewing ray without considering other effects of a real lens like vignetting, diffraction at the aperture or depth of field.

We evaluate the performance of the calibration process in simulation, starting with the simulated active display calibration process where the movement of the camera is known (Section 7.4.1). Next, the checkerboard calibration is evaluated using the GCM and BCM in simulation.

7.4.1 Simulated Active Display Camera Calibration

In this section, the DCM, calibrated with the active display calibration (Section 5.1), is evaluated in simulation. We use the same $3 \times 3 \times 4$ poses of the camera rig and the same wave numbers as in the real active display calibration (Section 7.3.1).

We model the display as an ideal target without any noise and discretization error, neither in spatial nor in brightness. The DCM is estimated and compared with the ground truth camera model using the CMDI in two settings: a flat and non-flat display surface. We will show that the estimated DCM with the active display calibration process is highly accurate. But as we will further demonstrate, in case of a non-flat display surface, the simplistic assumption of a flat display leads to a high inaccuracy of the estimated model.

Flat Display Surface

Using a flat display surface in simulation and a flat display surface function during estimation, the DCM can be perfectly estimated[1] for all settings, *no pane*, *perpendicular pane* or *angled pane*.

This is because of the way the dense measurements are generated from a display in front of the camera. Using simulated images without noise results in a perfect reconstruction of the viewing rays since a single ray per pixel is estimated and the spatial discretization of the camera does not affect the phase measurement. This results in a perfect monitor position estimate and thus, as the translations of the target are also known (not estimated), in a perfect ray estimation. The viewing rays are perfectly estimated even if a simulated glass pane is placed in front of the camera as we estimate one ray per pixel and the viewing ray variation within a single pixel is negligible. This underscores the generality of the DCM.

[1] ‘Perfect’ estimation means errors close to machine precision. That is the reason why we have not presented the CMDI as they would show a difference of zero for every single pixel.

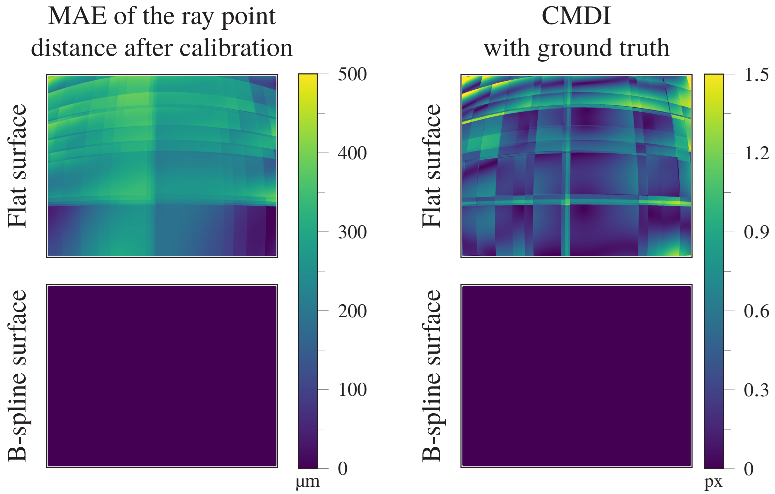


Figure 7.7: Simulation result for the active display calibration. This simulation uses a non-planar display. In the top row, the estimation results are shown assuming a flat surface, in the bottom row using a B-spline surface. Even though the flat surface assumption shows a maximum MAE of around 500 µm, the resulting pixel error is more than 1.5 px in certain areas. Using a B-spline surface shows a perfect fit with the ground truth.

Non-flat Display Surface

In this section, we focus our evaluation on the influence of a non-planar display. We answer the question of how a non-planar display affects the estimation errors of the DCM. In simulation, we model the non-flat display surface as a uniform B-spline. The parameters of the control points are taken from the real-world data of the calibration benchmark in Section 7.5. We estimate the camera model in two settings: First, we assume the display to be flat and second, we use a B-spline with the same number of control points as in simulation to model the display surface (see Section 5.1.3). For comparison, we plot the MAE of the ray point distance after calibration and the CMDI with the ground truth camera model used for simulation (see Section 6.2). The MAE of the ray point distance is used as a cost function to estimate the DCM. The results are shown in Fig. 7.7.

Even though the flat surface assumption shows a maximum MAE of around 500 µm, the resulting maximum CMDI is around 1.5 px. Additionally, there are jumps in the CMDI. These jumps are located at the border of the display. This is because different display poses contribute to different viewing rays since the display is generally not visible in all camera pixels. The model violation of the display surface then leads to jumps as we are minimizing the shortest distance between the viewing rays and the measured display point.

The choice of a B-spline surface results in a perfect fit with no difference to the ground truth camera model. This goes to show, that the chosen residual described in Section 5.1.3 enables us to correctly estimate the display surface using a uniform B-Spline.

7.4.2 Simulated Checkerboard Calibration

Our goal is to evaluate the GCM and the BCM using the CMDI in both a central and a non-central version. When we refer to the central version, we append a '-C', when we refer to the non-central version, we append a '-NC' to GCM or BCM respectively (see Table 7.1).

For simulation we need various poses (board poses, rig poses and extrinsic camera poses) and a camera model (see Section 5.2). We use the *no pane* setting of the checkerboard calibration dataset (see Section 7.3.2) to extract these poses after a calibration run. We use this information to render ideal checkerboard images.

The simulated images are shown in Fig. 7.8. The first column shows some recorded images of the real recorded sequence, the second one the ideal simulated images and the third column the detected checkerboards used for calibration. To generate the *perpendicular pane* and *angled pane* settings in simulation, we place an infinite plane in front of the middle camera at roughly the same location as in the real experiment.

All three settings are calibrated with the GCM camera model for the left and right cameras and the four camera models GCM-C, GCM-NC, BCM-C and BCM-NC for the middle camera. The estimated result is then compared with the ground truth camera model, using the CMDI to assess the performance.

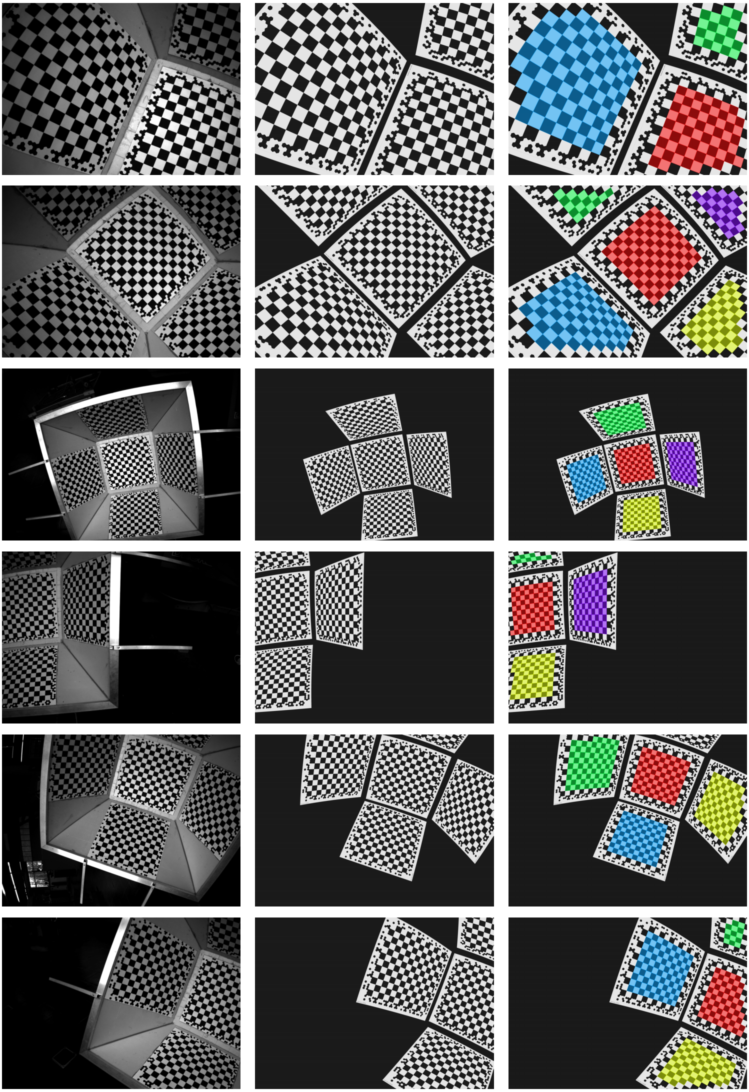


Figure 7.8: Checkerboard simulation input and output. The left column shows the recorded camera image which forms the basis for simulation, the middle shows the corresponding simulated camera image and the right column shows the detected checkerboard corners.

The evaluation results are shown only for the middle camera as this is the camera which is most affected by the pane. Despite that, the checkerboard calibration is performed with all three cameras jointly. This makes the result more accurate since the checkerboard poses can be estimated more accurately as multiple cameras at the same rig pose detect the same checkerboard.

We start with the discussion of the evaluation results of the GCM followed by those of the BCM.

Global Camera Model

The calibration results of the GCM are depicted in Fig. 7.9. Regarding the *no pane* setting, we expect an error close to zero as the simulated images were generated using a GCM without noise. The only remaining errors come from the spatial discretization of the camera image which result in some noise of the detected checkerboard corners. But, as can be seen in the reprojection error histogram (REH), that noise is very small and normally distributed with zero mean. The REH is point-symmetric with its maximum at zero. The same holds for the GCM-NC because the non-central model transitions into a central model when all non-central intrinsic parameters are set to zero. Not only is the RMS reprojection error very small but the CMDI is also almost zero. This can be seen in the CMDI plot as the difference is almost zero everywhere, meaning that the calibration algorithm converges to the correct minimum.

For the *perpendicular pane* setting, the RMS reprojection error for the central model increases slightly whereas for the non-central case it stays unchanged. As the viewing rays are displaced parallel when traveling through the glass pane, the camera model is no longer central. The displacement is dependent on the incident angle (see Fig. 4.4). Since the incident angle is nearly point-symmetric with respect to the optical axis, the chosen non-central camera model is a good fit. The CMDI shows almost no difference in the non-central case whereas the central model shows a significant difference. The difference is greater than 1 px at the image border even though the RMS reprojection error is 0.13 px. The RMS reprojection error is still small because the poses of the checkerboards and the camera rigs are optimized together with the camera model. If the model cannot cope with, e.g., non-central viewing rays, then the error may be partly compensated by changing the poses of the rig and the checkerboards.

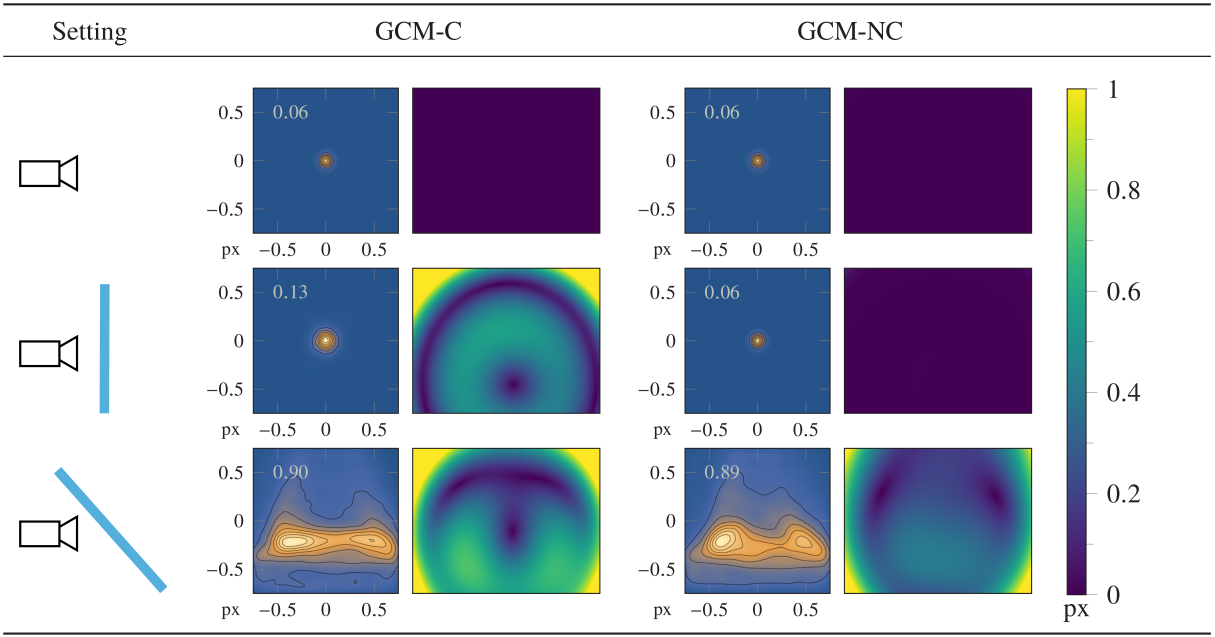


Figure 7.9: Evaluation results of the GCM in simulation. For the *no pane* setting, the estimate is perfect. For the *perpendicular pane* setting, the GCM-C shows significant errors especially at the image border whereas the GCM-NC is still an almost perfect fit. For the *angled pane* setting, both camera models are far away from the ground truth model.

The fact that the REH is still normally distributed makes the detection of modeling errors difficult. When the ground truth camera model is not known, it is hard to tell whether the increased error results from a higher checkerboard corner detector noise or from the wrong choice of camera model.

In the *angled pane* setting, both the central and the non-central camera model fail. The RMS reprojection error is significantly increased and the REH is not normally distributed. Even though the RMS reprojection error is almost the same for both models, the CMDI is slightly smaller for the non-central camera model.

Generic B-spline Distortion Camera Model

Now we move no to discuss the estimation results of the BCM-C and the BCM-NC.

The results of the BCM-C are comparable with those of the GCM-C. In the *no pane* setting, the RMS reprojection error is small and the CMDI between the ground truth camera model and the estimated one is close to zero. This shows that the BCM is capable of modeling a global camera model since a global model is used to simulate the rendered images. In the *perpendicular pane* setting, the RMS reprojection error increases slightly whereas the CMDI increases a lot. Considering the results of the *angled pane* setting, the RMS reprojection error conveys the impression that the BCM-C is superior to the GCM-C. However, the CMDIs between the camera models and the ground truth model show that the opposite is true. The higher number of parameters of the BCM-C combined with the simultaneous estimation of the poses leads to smaller reprojection errors but the estimated model is still worse.

For the BCM-NC, the reprojection errors are very small in all three settings and the CMDI to the ground truth model is almost zero. This means the optimization algorithm always converges to the correct global minimum and the non-central part induced by the glass pane can be modeled with a uniform B-spline.

We further compare the GCM-NC and the BCM-NC for the *perpendicular pane* setting in more detail by magnifying the CMDIs (see Fig. 7.11). Using the GCM-NC leads to a smaller and more homogeneous error compared to the BCM-NC. The BCM-NC also performs worse at the image borders. This

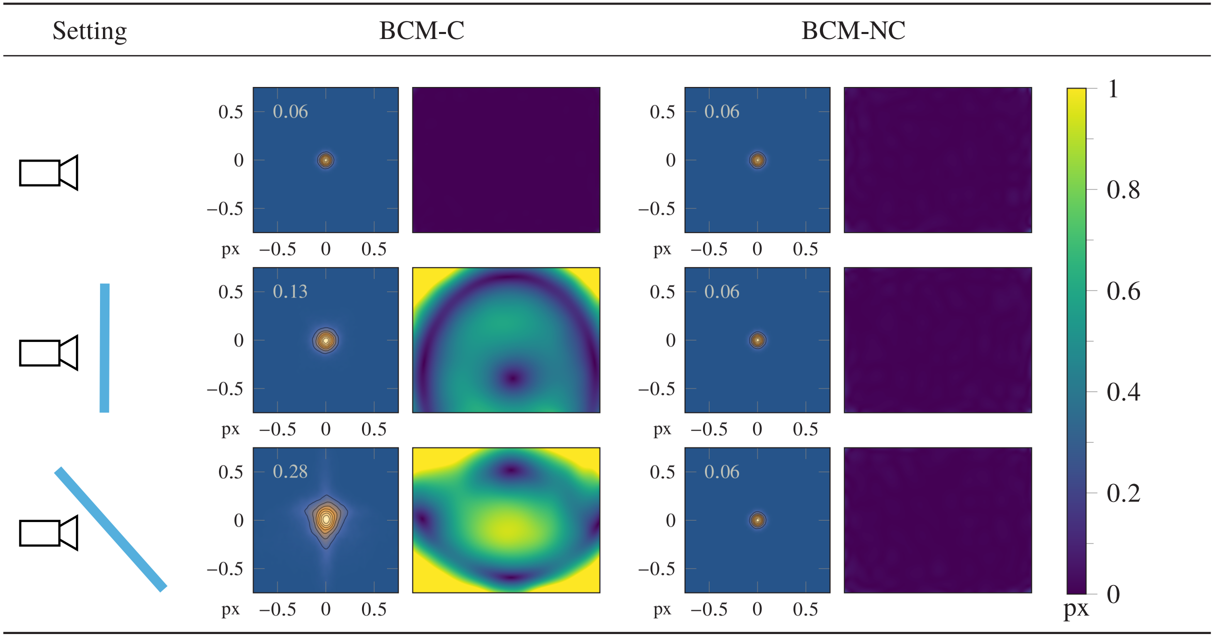


Figure 7.10: Evaluation results of the BCM in simulation. For the *no pane* setting, the estimate is perfect. For the *perpendicular pane* setting, the BCM-C shows significant errors especially at the image borders, whereas the GCM-NC is still an almost perfect fit. For the *angled pane* setting, the RMS reprojection error of the BCM-C is slightly bigger than before and the CMDI shows a significant difference, whereas the BCM-NC shows a low RMS reprojection error and a CMDI which is almost zero.

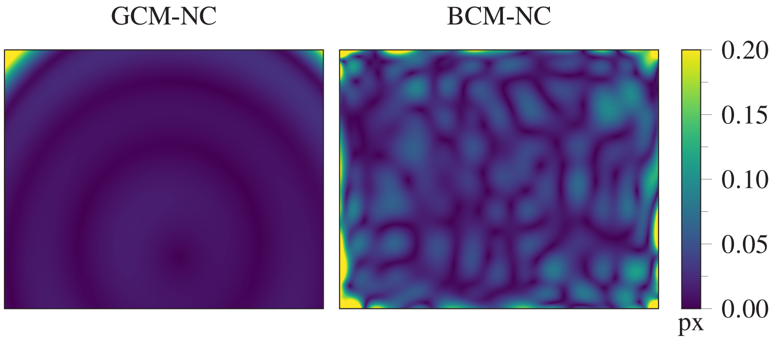


Figure 7.11: CMDI between the estimated GCM-NC (left image) / BCM-NC (right image) and the ground truth model in a more detailed view of the *perpendicular pane* setting.

is because B-splines have local support and the number of measurements decreases with the distance to the image borders.

Summary

The BCM-NC can handle all three settings, *no pane*, *perpendicular pane* and *angled pane*, with high accuracy. But due to the higher number of parameters compared to the GCM, more calibration images are needed. So it is advisable to choose the camera model with the least parameters which is sufficient for the given camera setup.

7.5 Calibration Benchmark

In order to evaluate different camera models in the real world, we propose a checkerboard camera calibration benchmark. For the benchmark, three things are needed: a checkerboard calibration dataset, the ground truth camera model and a measure to compare the ground truth camera model with the results of the checkerboard calibration.

For comparing camera models, we use the checkerboard calibration dataset described in Section 7.3.2 and the CMDI described in Section 6.2. In this section, we focus on how to generate the ground truth camera model.

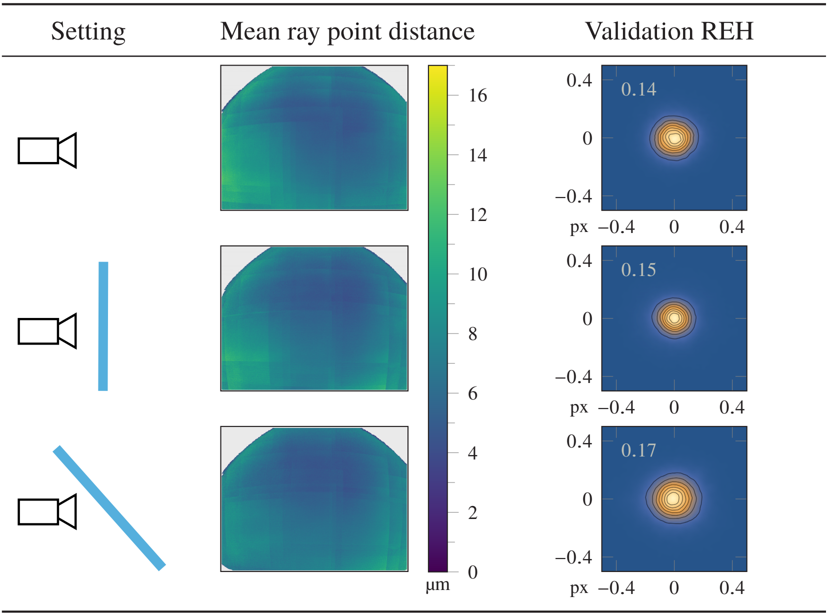


Figure 7.12: Visualization of the results of the active display calibration for the three different settings. The mean ray point distance in µm is shown in the middle column. As one viewing ray per pixel is estimated, the glass pane can be modeled perfectly. The validation checkerboard REHs are plotted on the right. They are normally distributed with an almost zero mean and a small RMS reprojection error. This proves the high accuracy of the estimated DCM using the active display calibration model. So it can be used as a ground truth camera model in the calibration benchmark.

The goal is to use a method which is more accurate than the checkerboard calibration and has fewer model assumptions. We achieve this by using the active display calibration (see Section 5.1) with the DCM. We have shown in simulation that this method is capable of estimating the camera model with very high accuracy (see Section 7.4.1). This is achieved by using the DCM and measuring the movement of the display with a linear positioning system. This enables us to estimate the display surface in order to increase the accuracy of the active display calibration to a level which makes it suitable for a ground truth calibration method.

The cost of using this method is its long measurement time and its reduced flexibility as the camera rig needs to be mounted on the linear positioning system.

For calibration, we use the dataset from Section 7.3.1. The DCM and the display surface are estimated jointly. The display surface is modeled as a uniform B-spline of order four with 14 × 7 control points.

To validate the resulting DCM, we use the recorded checkerboard calibration sequence (see Section 7.3.2), fix the camera model, estimate the checkerboard and rig poses and plot the REH. We expect the resulting REH to be normally distributed with a small RMS reprojection error.

As we need a continuous camera model for the checkerboard calibration, we fit the non-central generic B-spline distortion camera model (see Section 4.3) of order four and with 30 × 20 control points to the discrete viewing rays. The high number of control points is chosen in order to keep the fitting error almost zero. The results are shown in Fig. 7.12.

The mean ray point distance plot looks quite similar for all three settings, *no pane*, *perpendicular pane* and *angled pane*. The mean ray point distance is very small with a maximum value of about 17 µm. This shows that the DCM can be estimated correctly with the active display calibration process. A closer look does reveal some jumps in the error. We think that these come from the measured translation of the milling machine as the error is in the micrometer range. The validation REH is normally distributed with a small standard deviation and almost zero mean. The RMS reprojection error ranges from 0.14 to 0.17. This shows that the DCM we obtained from the active display calibration is suitable to be used as ground truth.

We further analyse the influence of the non-planar display surface. The estimated display surface is shown in Fig. 7.13. The distance between the highest and the lowest point is 1.9 mm for a display size of roughly 70 cm × 39 cm. The highest point is in the middle of the display which is reasonable as the display is placed horizontally in the milling machine and the stand is mounted in the middle of the display. To see the effect of the display surface model, we estimate the DCM using a flat display and validate that model with the checkerboard sequence. These results are compared with the non-flat display results (see Fig. 7.14). The maximum error of the mean ray point distance increases from 17 µm to 400 µm. Also, the validation REH shows a significantly higher standard

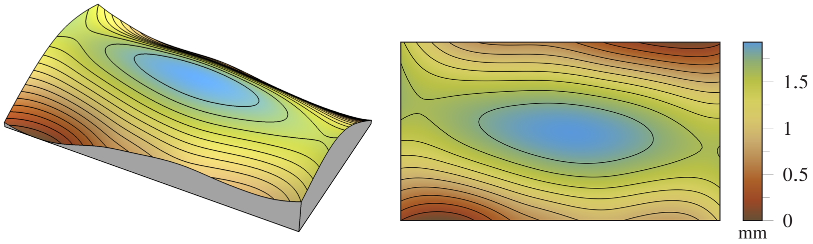


Figure 7.13: Visualization of the display surface, on the left in 3D and on the right as a contour plot. The distance between the highest and the lowest point is 1.9 mm for a display size of 70 cm × 39 cm.

deviation emphasizing the necessity of a non-planar display model for a highly accurate estimate of the DCM.

To sum up, we choose the active display calibration and estimate the display surface using a uniform B-spline to get a camera model which is able to serve as ground truth. The camera model is validated with the checkerboard calibration sequence. In the next section, we will use this benchmark to evaluate different camera models using the checkerboard calibration process.

7.6 Experimental Evaluation

In this section, the GCM and the BCM, estimated with the checkerboard calibration process, are evaluated in real world. We use two different camera setups: The first one uses the *triple camera setup* and the calibration benchmark to evaluate the performance (Section 7.6.1). The second one shows the performance benefits of the BCM over the GCM when calibrating the cameras of our experimental vehicle (Section 7.6.2).

7.6.1 Triple Camera Setup

To assess the accuracy of the GCM and the BCM calibrated with the checkerboard calibration process, we use the checkerboard calibration benchmark

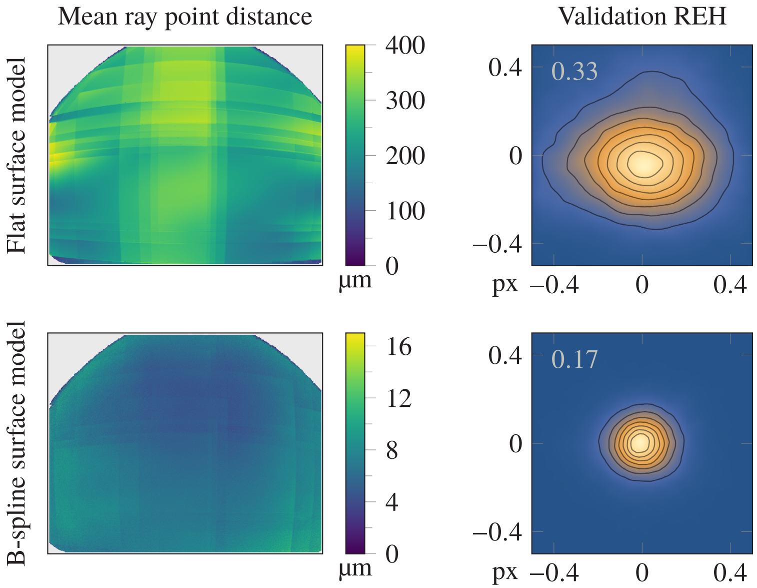


Figure 7.14: Comparison of the estimated DCM assuming a flat display (top row) and a non-planar display (bottom row) using the *angled pane* setting. On the left, the mean ray point distance is plotted and on the right the validation REH is plotted. The introduced errors when using a planar display surface are so huge that this model would not be accurate enough to serve as ground truth.

described in Section 7.5 and all three settings, *no pane*, *perpendicular pane* and *angled pane*.

To determine the camera model parameters, we jointly estimate the checkerboard poses, the rig poses, the extrinsic camera poses and the intrinsic camera parameters of each camera. Even though all benchmark sequences are synchronized by an external trigger, we did notice a significant increase in the RMS reprojection error and a bigger CMDI, independent of the camera model, if we estimate one rig pose for all cameras per recorded image and one extrinsic camera pose per camera. Both errors are reduced significantly if we estimate one rig pose per camera image. We think that these observed differences result from different trigger delays arising from different processing times in each camera.

We start with the discussion of the evaluation results of the GCM followed by those of the BCM.

Global Camera Model

The calibration results of the GCM-C and the GCM-NC are as follows (see Fig. 7.15): We see a similar pattern as in simulation for the *no pane* and *perpendicular pane* settings. Without a glass pane, the central and non-central models are almost equal in performance. Only at the borders of the image, the performance of the non-central model is slightly better as the lens itself is probably non-central due to the large FOV. For the *perpendicular pane* setting, the central model performs considerably worse while the non-central model still performs well. As in simulation, the REH of the central camera model is normally distributed. Thus, without a ground truth, it is hard to assess the performance of a calibration using only the RMS reprojection error.

In the third *angled pane* setting, both the GCM-C and the GCM-NC perform badly. The RMS reprojection error is significantly higher and the CMDI shows larger errors with an irregular pattern. The errors aren't as large as in simulation though (see Fig. 7.9) which probably comes from the fact that we additionally use the big checkerboards.

Generic B-spline Distortion Camera Model

Now, we discuss the results of the BCM-C and the BCM-NC (Fig. 7.16). The BCM-C shows a similar behavior as in simulation. In the *no pane* setting, the RMS reprojection error and the CMDI show only small deviations. In the *perpendicular pane* setting, the REH shows a normally distributed reprojection error and the CMDI shows a point-symmetric error around the optical axis. This is in good accordance with the simulation results (see Fig. 7.10). In the *angled pane* setting, the RMS reprojection error is smaller compared to the GCM but the CMDI shows larger errors. This comes from the fact that the flexibility of the BCM-C allows to estimate the poses wrongly in favour of a smaller reprojection error.

The BCM-NC performs equally well in all settings. As already shown in simulation, this camera model can cope with all kinds of distortions induced by the glass pane. However, due to the higher number of parameters, the accuracy

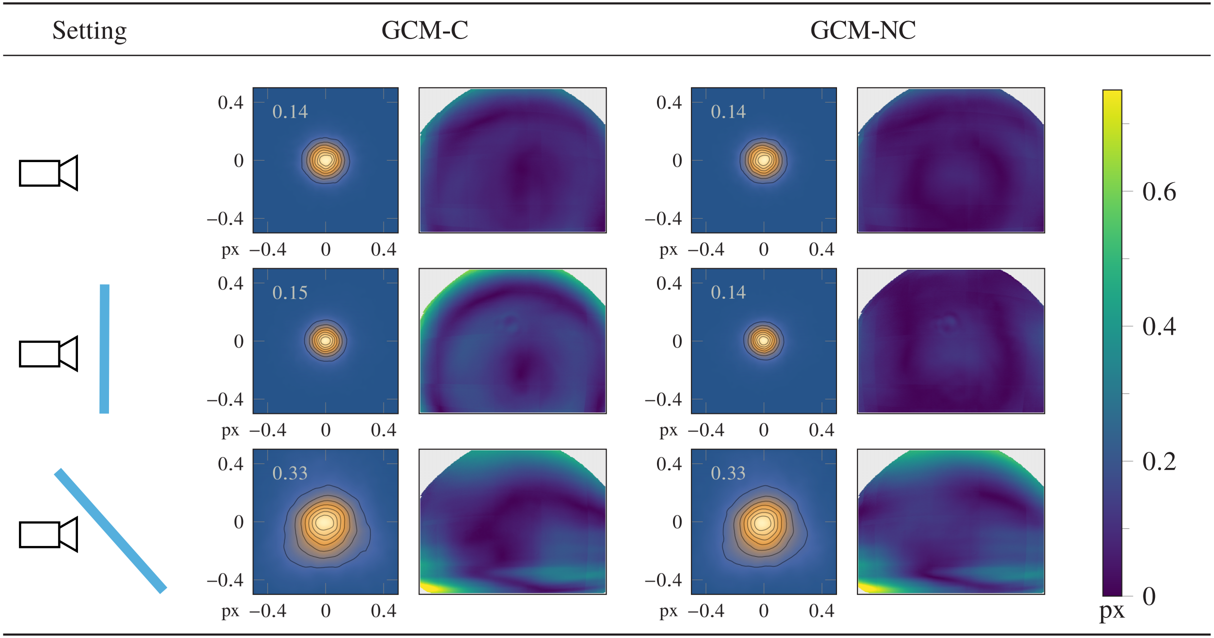


Figure 7.15: Evaluation results of the GCM using the calibration benchmark. In the *no pane* setting, the estimate is almost perfect. In the *perpendicular pane* setting, the REH is normally distributed and the CMDI shows the same patterns as in simulation. The central version cannot cope with the distortions induced by the perpendicular plane in front of the camera. For the last setting, *angled pane*, the REH is almost normally distributed but the RMS reprojection error is significantly increased. The CMDI shows huge differences at certain image locations and irregular patterns.

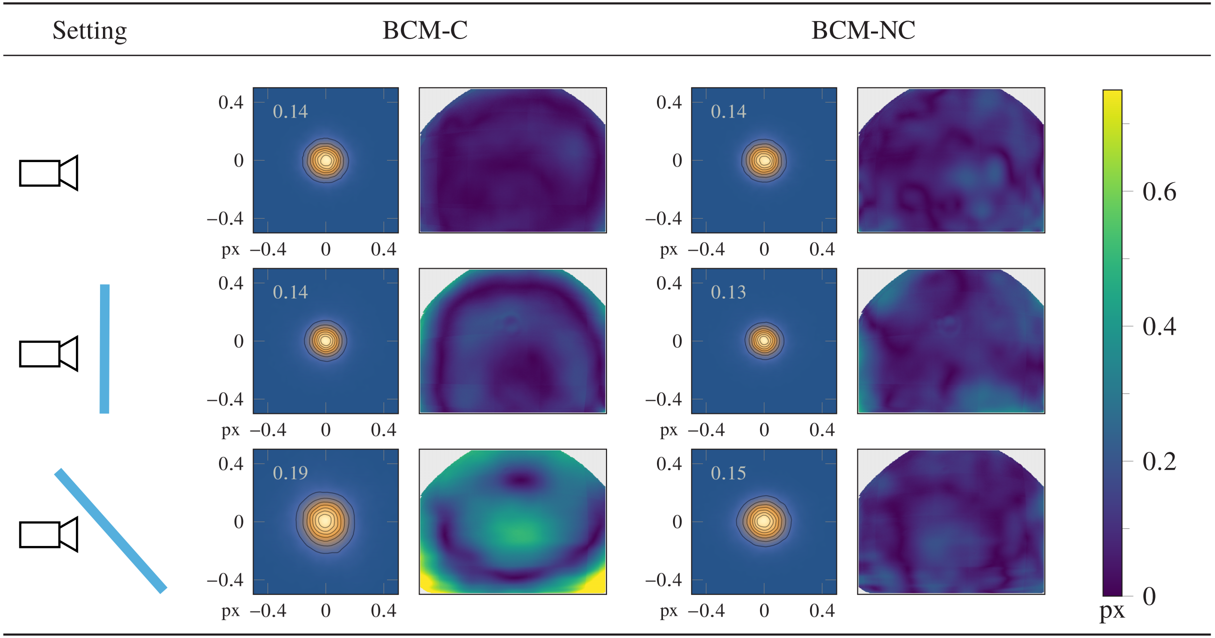


Figure 7.16: Evaluation results of the BCM using the calibration benchmark. The BCM-C performance is very similar to the simulation results. In the *no pane* setting, the estimate is almost perfect. In the *perpendicular pane* setting, the REH is normally distributed and the CMDI shows more or less the same patterns as in simulation. For the last setting, *angled pane*, the REH is still normally distributed but the CMDI is huge. The BCM-NC performance is very similar in each setting displaying small RMS reprojection errors and small CMDIs.

of the estimated BCM-NC is not quite as good as for the GCM-NC in the simple setting. This results in less accurate intrinsic camera parameters compared to the GCM when using the same number of checkerboard corners.

In the following section, we show the benefits of using the BCM-NC in a real-world calibration scenario.

7.6.2 Experimental Vehicle Calibration

In this section, a real-world camera calibration is performed using our experimental car. We demonstrate that the checkerboard calibration process scales to a real-world setup and analyze the effect of choosing proper camera models. We use six cameras in total: one color camera and a stereo pair pointing forward, one camera each to the left and to the right side and one to the rear. All cameras have a resolution ranging from 2.8 Mpx to 7.2 Mpx and a FOV of at least 110°. The used cameras and attached lenses are depicted in Fig. 7.17.

We use two different checkerboard targets for calibration: The first setup uses five boards arranged in a truncated pyramid (see Fig. 7.4) and the second board setup uses four 1 m × 2 m boards placed around the experimental car. We use the image sequence of both setups to jointly optimize the board, frame and rig poses and the camera model parameters.

The cameras pointing forward are mounted behind the windshield, the one pointing backwards behind the rear window and the cameras to the side are mounted uncovered. We compare two different settings: In the first setting, we choose GCM-NC (see Section 7.1) for all six cameras. In the second setting, we switch the four camera models for those cameras that are mounted behind a window to BCM-NC. The REHs are shown in Fig. 7.18. As can be seen, the reprojection errors are significantly smaller when using BCM-NC instead of GCM-NC. This is also true for the cameras not mounted behind a window even though the same camera models are used. The GCM-NC cannot cope with the distortions from a window. As all poses and the camera model parameters are estimated jointly, the modeling error results in wrongly estimated poses. This also affects the camera models not behind a window resulting in higher reprojection errors for all cameras.

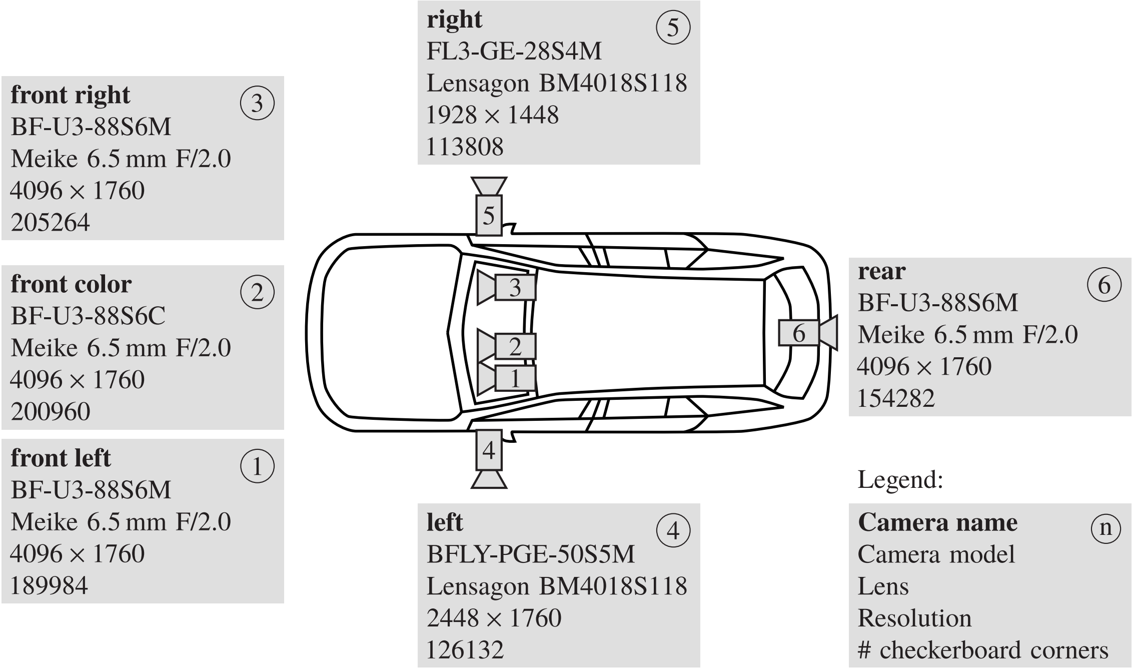


Figure 7.17: Camera setup used for the experimental vehicle calibration.

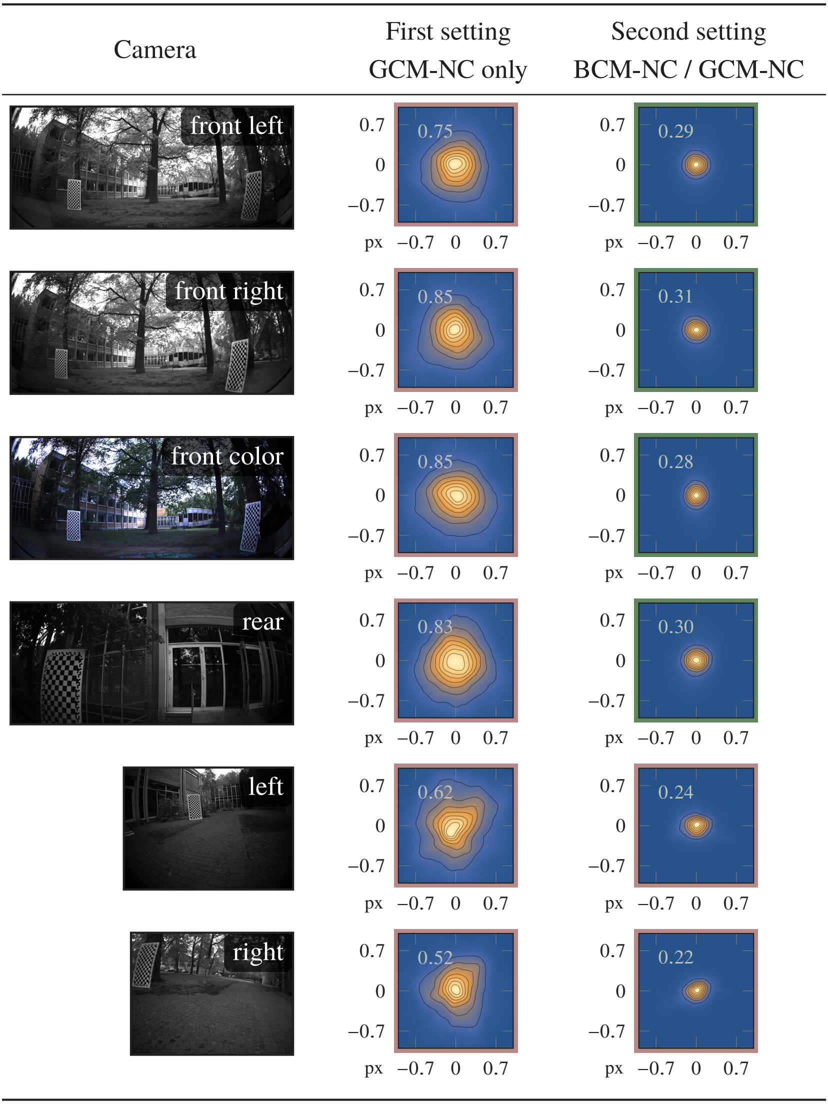


Figure 7.18: Example images for all cameras and REHs for two different settings. GCM-NC is boxed in red and BCM-NC is boxed in green. In the first setting, we use the GCM-NC only. In the second setting, BCM-NC is used for all cameras behind a window. The second setting outperforms the first one reducing the RMS reprojection error by a factor of two to three for all cameras.

7.7 Conclusion

As seen in the real-world experiment as well as in simulation, the BCM-NC can handle various lens distortions and non-central optical systems resulting from a glass pane in front of the camera and the BCM-NC can be accurately estimated using a checkerboard calibration process. The experimental vehicle calibration demonstrates a real-world application that hugely benefits from using the BCM-NC over GCM-NC for cameras behind a window. But one need to keep in mind that due to the higher number of parameters of this camera model, more checkerboard detections are needed to obtain an accurate estimate.

Hence, for good calibration results, it is key to select the right camera model which is in compliance with the whole optical system consisting not only of the lens but also of windshields or similar optical sources of distortion. In general, choosing an unsuitable camera model for one camera will negatively affect the calibration of all other camera models. If one is certain that a global camera model is sufficient, then it is usually preferred over a local model due to its lower number of parameters. But if in doubt, it is preferable to use a local camera model and longer calibration sequences.

Another finding is that the REH and especially the RMS reprojection error is not always a good measure to assess the estimation quality of a camera model. If the camera model is not selected in compliance with the used optical system, then the REH may be normally distributed and the RMS reprojection error small but the difference to the true camera model may still be huge.

8 Summary and Outlook

In this thesis, we presented a novel non-central B-spline distortion camera model belonging to the class of local camera models. A camera model maps pixel positions to a viewing ray described by a base point and a direction vector. The direction vector was modeled by an ideal equidistant angle projection model combined with a distortion function based on uniform B-splines. The base point of the viewing ray was modeled as a displacement vector perpendicular to the direction vector. The displacement vector field was also described by a uniform B-spline formulated on the tangent plane of the direction vector field.

We demonstrated that this camera model can be estimated in a multi-camera setup by an easy-to-use calibration process based on checkerboard targets. To generate the input data of the calibration process, one simply records a sequence of images in which the checkerboard targets are viewed from different orientations. To estimate the camera model parameters and the translation between the cameras, a non-linear least squares problem is formulated and solved. As a residuum function, we used the reprojection error with the forward camera model, which projects a point in 3D onto the image. We showed how to derive the Jacobian matrix for camera models where only an implicit formulation of the forward camera model is available, which is often the case for non-central models. To increase the robustness of the estimate in areas where the density of measurements in the image is low, a prior which can be efficiently integrated into the problem formulation based on the smoothness of uniform B-splines was added. We showed in simulation that a glass pane mounted in front of a camera leads to distortions - distortions which can be handled by the proposed non-central B-spline distortion camera model.

To evaluate this novel camera model not only in simulation, we developed a calibration benchmark consisting of an experimental setup, a ground truth camera model and a measure to compare camera models with each other. For the experimental setup, three cameras pointing in the same direction were used. For one of the cameras, a glass pane was mounted in different angles to simulate the effect of a windshield. The ground truth camera model was generated by an active display camera calibration process. For this purpose, the experimental setup was mounted on a linear positioning system facing in the direction of a monitor displaying different sinusoidal fringe patterns. These patterns were used to estimate the positions of 2D points on the display which were then used to determine a discrete camera model. We showed that the monitor cannot be assumed to be flat and therefore modeled the display surface as a uniform B-spline. The surface and the viewing rays were jointly estimated by a non-linear least squares solver. The rays were parametrized by the base point and the line direction, and a local parametrization was proposed to speed up the estimate and increase its robustness. This resulted in a highly accurate camera model which served as ground truth verified in simulation. The measure for comparing different camera models was based on projecting points in world space to the image space of both camera models and then calculating the difference between these two points.

We used the calibration benchmark to evaluate a global and the B-spline distortion camera model, in both a central and a non-central formulation. In the case of cameras mounted behind an angled glass pane, the non-central B-spline distortion model outperforms all others. We also demonstrated the benefits of using this camera model when calibrating our experimental vehicle equipped with a surround-view camera system consisting of six cameras.

All of the experiments showed that it is essential to choose a camera model which fits the used camera system and that the widely utilized RMS reprojection error cannot be used as a quality measure for checkerboard camera calibration if the wrong camera model was chosen.

This brings us to the open question of how to measure the quality of an estimated camera model when no ground truth is available. A future field of research could be to investigate the use of additional sensors such as range-sensors and inertial measurements to assess the quality of the estimated camera model.

Another direction would be to not use artificial targets but instead to calibrate the sensor setup with natural scenes in order to further simplify the calibration process. To assess the performance of such estimated camera models, the proposed benchmark can provide a ground truth camera model.

Bibliography

[ATR10] A. Agrawal, Y. Taguchi, and S. Ramalingam, "Analytical forward projection for axial non-central dioptric and catadioptric cameras," in *Computer Vision - ECCV*, vol. 6313. Berlin, Germany: Springer Verlag, 2010, pp. 129–143, DOI: http://dx.doi.org/10.1007/978-3-642-15558-1_10.

[ATR11] A. Agrawal, Y. Taguchi, and S. Ramalingam, "Beyond Alhazen's problem: Analytical projection model for non-central catadioptric cameras with quadric mirrors," in *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 2011, pp. 2993–3000, DOI: http://dx.doi.org/10.1109/CVPR.2011.5995596.

[BART13] F. Bergamasco, A. Albarelli, E. Rodola, and A. Torsello, "Can a fully unconstrained imaging model be applied effectively to central cameras?" in *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, 2013, pp. 1391–1398, DOI: http://dx.doi.org/10.1109/CVPR.2013.183.

[BBV01] C. Bräuer-Burchardt and K. Voss, "A new algorithm to correct fish-eye- and strong wide-angle-lens-distortion from single images," in *Proceedings - IEEE International Conference on Image Processing*, vol. 1, Thessaloniki, Greece, 2001, pp. 225–228, DOI: http://dx.doi.org/10.1109/icip.2001.958994.

[BCG$^+$17] F. Bergamasco, L. Cosmo, A. Gasparetto, A. Albarelli, and A. Torsello, "Parameter-Free Lens Distortion Calibration of Central Cameras," in *Proceedings - IEEE International Conference on Computer Vision*, Venice, Italy, 2017, pp. 3867–3875, DOI: http://dx.doi.org/10.1109/ICCV.2017.415.

[Bec25] C. Beck, "Apparatus to photograph the whole sky," in *Journal of Scientific Instruments*, vol. 2, no. 4, pp. 135–139, 1925, DOI: http://dx.doi.org/10.1088/0950-7671/2/4/305.

[Bjö96] Å. Björck, *Numerical Methods for Least Squares Problems*, ser. Other Titles in Applied Mathematics. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1996. DOI: http://dx.doi.org/10.1137/1.9781611971484.

[Bjö15] Å. Björck, *Numerical Methods in Matrix Computations*, ser. Texts in Applied Mathematics. Basel, Switzerland: Springer International Publishing, 2015, vol. 59, DOI: http://dx.doi.org/10.1007/978-3-319-05089-8.

[BL93] A. Basu and S. Licardie, "Modeling fish-eye lenses," in *Proceedings - IEEE International Conference on Intelligent Robots and Systems*, Yokohama, Japan, 1993, pp. 1822–1828, DOI: http://dx.doi.org/10.1109/iros.1993.583883.

[BL95] A. Basu and S. Licardie, "Alternative models for fish-eye lenses," in *Pattern Recognition Letters*, vol. 16, no. 4, pp. 433–441, 1995, DOI: http://dx.doi.org/10.1016/0167-8655(94)00115-J.

[BNPR19] S. Barone, P. Neri, A. Paoli, and A. V. Razionale, "Flexible calibration of a stereo vision system by active display," in *Procedia Manufacturing*, vol. 38. Limerick, Ireland: Elsevier, 2019, pp. 564–572, DOI: http://dx.doi.org/10.1016/j.promfg.2020.01.071.

[BNPR20] S. Barone, P. Neri, A. Paoli, and A. V. Razionale, "3D acquisition and stereo-camera calibration by active devices: A unique structured light encoding framework," in *Optics and Lasers in Engineering*, vol. 127, p. N/A, 2020, DOI: http://dx.doi.org/10.1016/j.optlaseng.2019.105989.

[BP02] H. Bakstein and T. Pajdla, "Panoramic mosaicing with a 180° field of view lens," in *Proceedings - IEEE Workshop on Omnidirectional Vision*, Copenhagen, Denmark, 2002, pp. 60–67, DOI: http://dx.doi.org/10.1109/OMNVIS.2002.1044492.

[Bra00] G. Bradski, "The OpenCV Library," in *Dr. Dobb's Journal of Software Tools*, vol. 25, pp. 120–125, 2000, DOI: http://dx.doi.org/10.1111/0023-8333.50.s1.10.

[Bro66] D. Brown, "Decentering Distortion of Lenses," in *Photometric Engineering*, vol. 32, no. 3, pp. 444–462, 1966. [Online]. Available: https://ci.nii.ac.jp/naid/10022411406/ last retrieved 2020-09-22.

[CF05] D. Claus and A. W. Fitzgibbon, “A rational function lens distortion model for general cameras,” in *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, San Diego, CA, USA, 2005, pp. 213–219, DOI: http://dx.doi.org/10.1109/CVPR.2005.43.

[CLSC92] G. Champleboux, S. Lavallee, P. Sautot, and P. Cinquin, “Accurate calibration of cameras and range imaging sensor: The NPBS method,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, Nice, France, 1992, pp. 1552–1557, DOI: http://dx.doi.org/10.1109/robot.1992.220031.

[DF01] F. Devernay and O. Faugeras, “Straight lines have to be straight,” in *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 2001, DOI: http://dx.doi.org/10.1007/PL00013269.

[DMW10] A. K. Dunne, J. Mallon, and P. F. Whelan, “Efficient generic calibration method for general cameras with single centre of projection,” in *Computer Vision and Image Understanding*, vol. 114, no. 2, pp. 220–233, 2010, DOI: http://dx.doi.org/10.1016/j.cviu.2009.05.005.

[Duc77] J. Duchon, “Splines minimizing rotation-invariant semi-norms in Sobolev spaces,” in *Constructive Theory of Functions of Several Variables*, vol. 571. Berlin, Germany: Springer Verlag, 1977, pp. 85–100, DOI: http://dx.doi.org/10.1007/bfb0086566.

[Fit01] A. W. Fitzgibbon, “Simultaneous linear estimation of multiple view geometry and lens distortion,” in *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, Kauai, HI, USA, 2001, pp. I–125 – I–132, DOI: http://dx.doi.org/10.1109/cvpr.2001.990465.

[GD00] C. Geyer and K. Daniilidis, “A unifying theory for central panoramic systems and practical implications,” in *Computer Vision - ECCV*, vol. 1843. Berlin, Germany: Springer Verlag, 2000, pp. 445–461, DOI: http://dx.doi.org/10.1007/3-540-45053-x_29.

[Gen06] D. B. Gennery, “Generalized camera calibration including fish-eye lenses,” in *International Journal of Computer Vision*, vol. 68, no. 3, pp. 239–266, 2006, DOI: http://dx.doi.org/10.1007/s11263-006-5168-1.

[Gen11] J. Geng, “Structured-light 3D surface imaging: a tutorial,” in *Advances in Optics and Photonics*, vol. 3, no. 2, pp. 128–160, 2011, DOI: http://dx.doi.org/10.1364/aop.3.000128.

[GMCS12] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, “Automatic camera and range sensor calibration using a single shot,” in *Proceedings - IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, 2012, pp. 3936–3943, DOI: http://dx.doi.org/10.1109/ICRA.2012.6224570.

[GN01] M. D. Grossberg and S. K. Nayar, “A general imaging model and a method for finding its parameters,” in *Proceedings - IEEE International Conference on Computer Vision*, vol. 2, pp. 108–115, 2001, DOI: http://dx.doi.org/10.1109/iccv.2001.937611.

[GN05] M. D. Grossberg and S. K. Nayar, “The raxel imaging model and ray-based calibration,” in *International Journal of Computer Vision*, vol. 61, pp. 119–137, 2005, DOI: http://dx.doi.org/10.1023/B:VISI.0000043754.56350.10.

[GN09] N. Gonçalves and A. C. Nogueira, “Projection through quadric mirrors made faster,” in *Proceedings - IEEE International Conference on Computer Vision Workshops*, Kyoto, Japan, 2009, pp. 2141–2148, DOI: http://dx.doi.org/10.1109/ICCVW.2009.5457545.

[Gon10] N. Gonçalves, “On the reflection point where light reflects to a known destination on quadratic surfaces,” in *Optics Letters*, vol. 35, no. 2, pp. 100–102, 2010, DOI: http://dx.doi.org/10.1364/ol.35.000100.

[GTK88] K. D. Gremban, C. E. Thorpe, and T. Kanade, “Geometric Camera Calibration using Systems of Linear Equations,” in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, Philadelphia, PA, USA, 1988, pp. 562–567, DOI: http://dx.doi.org/10.1109/ROBOT.1988.12111.

[Han11] T. Hanning, *High Precision Camera Calibration*. Wiesbaden, Germany: Vieweg+Teubner Verlag, Springer, 2011. DOI: http://dx.doi.org/10.1007/978-3-8348-9830-2.

[Hei00] J. Heikkilä, “Geometric camera calibration using circular control points,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1066–1077, 2000, DOI: http://dx.doi.org/10.1109/34.879788.

[HS96] J. Heikkila and O. Silvén, “Calibration procedure for short focal length off-the-shelf CCD cameras,” in *Proceedings - IEEE International Conference on Pattern Recognition*, vol. 1, Vienna, Austria, 1996, pp. 166–170, DOI: http://dx.doi.org/10.1109/ICPR.1996.546012.

[HSK16] H. Hoppe, F. Seebacher, and M. Klemm, "Nicht-modellbasierte kalibrierung von kameras mit monitoren," in *Bildverarbeitung für die Medizin*. Berlin, Germany: Springer Verlag, 2016, pp. 50–55, DOI: http://dx.doi.org/10.1007/978-3-662-49465-3_11.

[HZ04] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2004. DOI: http://dx.doi.org/10.1017/CBO9780511811685.

[HZA13] L. Huang, Q. Zhang, and A. Asundi, "Flexible camera calibration using not-measured imperfect target," in *Applied Optics*, vol. 52, no. 25, pp. 6278–6286, 2013, DOI: http://dx.doi.org/10.1364/AO.52.006278.

[JQ05] G. Jiang and L. Quan, "Detection of concentric circles for camera calibration," in *Proceedings - IEEE International Conference on Computer Vision*, vol. 1, Beijing, China, 2005, pp. 333–340, DOI: http://dx.doi.org/10.1109/iccv.2005.73.

[KB00] J. J. Kumler and M. L. Bauer, "Fish-eye lens designs and their relative performance," in *Proceedings - Current Developments in Lens Design and Optical Systems Engineering*, vol. 4093, no. 24. San Diego, CA, USA: SPIE, oct 2000, pp. 360 – 369, DOI: http://dx.doi.org/10.1117/12.405226.

[KB06] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1335–1340, 2006, DOI: http://dx.doi.org/10.1109/TPAMI.2006.153.

[KGM16] B. Khomutenko, G. Garcia, and P. Martinet, "An Enhanced Unified Camera Model," in *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 137–144, jan 2016, DOI: http://dx.doi.org/10.1109/LRA.2015.2502921.

[Kui99] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace and Virtual Reality*. Princeton, NJ, USA: Princeton University Press, 1999. DOI: http://dx.doi.org/10.2307/j.ctvx5wc3k.

[LM02] L. Lucchese and S. K. Mitra, "Using saddle points for subpixel feature detection in camera calibration targets," in *Proceedings - IEEE Asia-Pacific Conference on Circuits and Systems*, vol. 2, Denpasar, Bali, Indonesia, 2002, pp. 191–195, DOI: http://dx.doi.org/10.1109/APCCAS.2002.1115151.

[LRKB19] T. Luhmann, S. Robson, S. Kyle, and J. Boehm, *Close-Range Photogrammetry and 3D Imaging*, 3rd ed., ser. De Gruyter STEM. Berlin, Germany: De Gruyter, 2019. DOI: http://dx.doi.org/10.1515/9783110607253.

[LT88] R. K. Lenz and R. Y. Tsai, "Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 713–720, 1988, DOI: http://dx.doi.org/10.1109/34.6781.

[Lub15] A. Luber, "Ein generisches Abbildungsmodell für Stereokamerasysteme – Modellierung, Kalibrierung, Evaluierung und Anwendung," doctoral dissertation, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Berlin, Germany, 2015, DOI: http://dx.doi.org/10.18452/17114.

[MA13] P. Miraldo and H. Araujo, "Calibration of smooth camera models," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2091–2103, 2013, DOI: http://dx.doi.org/10.1109/TPAMI.2012.258.

[MAQ11] P. Miraldo, H. Araujo, and J. Queiró, "Point-based calibration using a parametric representation of the general imaging model," in *Proceedings - IEEE International Conference on Computer Vision*, Barcelona, Spain, 2011, pp. 2304–2311, DOI: http://dx.doi.org/10.1109/ICCV.2011.6126511.

[MBK81] H. A. Martins, J. R. Birk, and R. B. Kelley, "Camera models based on data from two calibration planes," in *Computer Graphics and Image Processing*, vol. 17, no. 2, pp. 173–180, oct 1981, DOI: http://dx.doi.org/10.1016/0146-664X(81)90024-1.

[McL00] P. F. McLauchlan, "Gauge independence in optimization algorithms for 3D vision," in *IWVA: Vision Algorithms: Theory and Practice*, vol. 1883. Berlin, Germany: Springer Verlag, 2000, pp. 183–199, DOI: http://dx.doi.org/10.1007/3-540-44480-7_12.

[MCM04] L. Ma, Y. Chen, and K. L. Moore, "Rational radial distortion models of camera lenses with analytical solution for distortion correction," in *International Journal of Information Acquisition*, vol. 01, no. 02, pp. 135–147, 2004, DOI: http://dx.doi.org/10.1142/s0219878904000173.

[MCM06] L. Ma, Y. Chen, and K. L. Moore, “Analytical piecewise radial distortion model for precision camera calibration,” in *IEE Proceedings - Vision, Image and Signal Processing*, vol. 153, no. 4. London, UK: IET, 2006, pp. 468–474, DOI: http://dx.doi.org/10.1049/ip-vis:20045035.

[MH03] X. Meng and Z. Hu, “A new easy camera calibration technique based on circular points,” in *Pattern Recognition*, vol. 36, no. 5, pp. 1155–1164, 2003, DOI: http://dx.doi.org/10.1016/S0031-3203(02)00225-X.

[MR07] C. Mei and P. Rives, “Single view point omnidirectional camera calibration from planar grids,” in *Proceedings - IEEE International Conference on Robotics and Automation*, Roma, Italy, 2007, pp. 3945–3950, DOI: http://dx.doi.org/10.1109/ROBOT.2007.364084.

[MW04] J. Mallon and P. F. Whelan, “Precise radial un-distortion of images,” in *Proceedings - IEEE International Conference on Pattern Recognition*, vol. 1, Cambridge, UK, 2004, pp. 18–21, DOI: http://dx.doi.org/10.1109/ICPR.2004.1333995.

[MW07] J. Mallon and P. F. Whelan, “Which pattern? Biasing aspects of planar calibration patterns and detection methods,” in *Pattern Recognition Letters*, vol. 28, no. 8, pp. 921–930, 2007, DOI: http://dx.doi.org/10.1016/j.patrec.2006.12.008.

[Ng91] E. Ng, “A Scheme for Handling Rank-Deficiency in the Solution of Sparse Linear Least Squares Problems,” in *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 5, pp. 1173–1183, sep 1991, DOI: http://dx.doi.org/10.1137/0912062.

[NW06] J. Nocedal and S. J. Wright, *Numerical Optimization*, ser. Operations Research and Financial Engineering (ORFE). New York, NY, USA: Springer Science & Business Media, 2006. DOI: http://dx.doi.org/10.1007/978-0-387-40065-5.

[PBSG12] L. Puig, J. Bermúdez, P. Sturm, and J. J. Guerrero, “Calibration of omnidirectional cameras in practice: A comparison of methods,” in *Computer Vision and Image Understanding*, vol. 116, no. 1, pp. 120–137, 2012, DOI: http://dx.doi.org/10.1016/j.cviu.2011.08.003.

[PL00] T. Papadopoulo and M. I. Lourakis, “Estimating the jacobian of the singular value decomposition: Theory and applications?” in *Computer Vision - ECCV*, vol. 1842. Berlin, Germany: Springer Verlag, 2000, pp. 554–570, DOI: http://dx.doi.org/10.1007/3-540-45054-8_36.

[Rap12] H. H. Rapp, *Reconstruction of Specular Reflective Surfaces using Auto-Calibrating Deflectometry*, ser. Schriftenreihe des Instituts für Mess- und Regelungstechnik, Karlsruher Institut für Technologie. Karlsruhe, Germany: KIT Scientific Publishing, 2012, no. 23, DOI: http://dx.doi.org/10.5445/KSP/1000032148, doctoral dissertation.

[Ros16] D. Rosebrock, *The Surface Model: An Uncertain Continuous Representation of the Generic Camera Model and Its Calibration*. Düren, Germany: Shaker, 2016.

[RSL05] S. Ramalingam, P. Sturm, and S. K. Lodha, “Towards complete generic camera calibration,” in *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, San Diego, CA, USA, 2005, pp. 1093–1098, DOI: http://dx.doi.org/10.1109/CVPR.2005.347.

[RSO13] A. Richardson, J. Strom, and E. Olson, “AprilCal: Assisted and repeatable camera calibration,” in *Proceedings - IEEE International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013, pp. 1814–1821, DOI: http://dx.doi.org/10.1109/IROS.2013.6696595.

[RW12] D. Rosebrock and F. M. Wahl, “Generic camera calibration and modeling using spline surfaces,” in *Proceedings - IEEE Intelligent Vehicles Symposium*, Alcala de Henares, Spain, 2012, pp. 51–56, DOI: http://dx.doi.org/10.1109/IVS.2012.6232156.

[SA96] S. Shah and J. K. Aggarwal, “Intrinsic parameter calibration procedure for a (high-distortion) fish-eye lens camera with distortion model and accuracy estimation,” in *Pattern Recognition*, vol. 29, no. 11, pp. 1775–1788, nov 1996, DOI: http://dx.doi.org/10.1016/0031-3203(96)00038-6.

[SAB02] J. Salvi, X. Armangué, and J. Batlle, “A comparative review of camera calibrating methods with accuracy evaluation,” in *Pattern Recognition*, vol. 35, no. 7, pp. 1617–1635, 2002, DOI: http://dx.doi.org/10.1016/S0031-3203(01)00126-1.

[Sch14] M. Schönbein, *Omnidirectional Stereo Vision for autonomous Vehicles*, ser. Schriftenreihe des Instituts für Mess- und Regelungstechnik, Karlsruher Institut für Technologie. Karlsruhe, Germany: KIT Scientific Publishing, 2014, no. 32, DOI: http://dx.doi.org/10.5445/KSP/1000046298, doctoral dissertation.

[SGN06] R. Swaminathan, M. D. Grossberg, and S. K. Nayar, “Non-single viewpoint catadioptric cameras: Geometry and analysis,” in *International Journal of Computer Vision*, vol. 66, pp. 211–229, 2006, DOI: http://dx.doi.org/10.1007/s11263-005-3220-1.

[SH11] K. H. Strobl and G. Hirzinger, “More accurate pinhole camera calibration with imperfect planar target,” in *Proceedings - IEEE International Conference on Computer Vision Workshops*, Barcelona, Spain, 2011, pp. 1068–1075, DOI: http://dx.doi.org/10.1109/ICCVW.2011.6130369.

[Shl14] J. Shlens, *A Tutorial on Principal Component Analysis*, apr 2014, *arXiv:1404.1100*. [Online]. Available: http://arxiv.org/abs/1404.1100 last retrieved 2020-09-22.

[SLPS20] T. Schöps, V. Larsson, M. Pollefeys, and T. Sattler, “Why Having 10,000 Parameters in Your Camera Model is Better Than Twelve,” in *Proceedings - IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2535–2544. [Online]. Available: http://arxiv.org/abs/1912.02908 last retrieved 2020-09-22.

[SM99] P. F. Sturm and S. J. Maybank, “On plane-based camera calibration: A general algorithm, singularities, applications,” in *Proceedings - IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Fort Collins, CO, USA, 1999, pp. 432–437, DOI: http://dx.doi.org/10.1109/CVPR.1999.786974.

[SMS06] D. Scaramuzza, A. Martinelli, and R. Siegwart, “A flexible technique for accurate omnidirectional camera calibration and structure from motion,” in *Proceedings - IEEE International Conference on Computer Vision Systems*, New York, NY, USA, 2006, pp. 45–52, DOI: http://dx.doi.org/10.1109/ICVS.2006.3.

[Sob70] I. Sobel, *Camera models and machine perception*. Stanford, CA, USA: Stanford University, 1970, doctoral dissertation.

[SR04] P. Sturm and S. Ramalingam, “A generic concept for camera calibration,” in *Computer Vision - ECCV*, vol. 3022. Berlin, Germany: Springer, 2004, pp. 1–13, DOI: http://dx.doi.org/10.1007/978-3-540-24671-8_1.

[SRT+11] P. Sturm, S. Ramalingam, J. P. Tardif, S. Gasparini, and J. Barreto, “Camera models and fundamental concepts used in geometric computer vision,” in *Foundations and Trends in Computer Graphics and Vision*, vol. 6, no. 1-2, pp. 1–183, 2011, DOI: http://dx.doi.org/10.1561/0600000023.

[Str15] T. Strauss, “Kalibrierung von Multi-Kamera-Systemen - Kombinierte Schätzung von intrinsischem Abbildungsverhalten der einzelnen Kameras und deren relativer Lage zueinander ohne Erfordernis sich überlappender Sichtbereiche,” doctoral dissertation, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, 2015, DOI: http://dx.doi.org/10.5445/IR/1000051877.

[Tsa87] R. Y. Tsai, “A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses,” in *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987, DOI: http://dx.doi.org/10.1109/JRA.1987.1087109.

[UDC18] V. Usenko, N. Demmel, and D. Cremers, “The double sphere camera model,” in *Proceedings - IEEE International Conference on 3D Vision*, Verona, Italy, jul 2018, pp. 552–560, DOI: http://dx.doi.org/10.1109/3DV.2018.00069.

[VG13] C. F. Van Loan and G. H. Golub, *Matrix Computations*, 4th ed. Baltimore, MD, USA: Johns Hopkins University Press, 2013.

[VT05] C. Y. Vincent and T. Tjahjadi, “Multiview camera-calibration framework for nonparametric distortions removal,” in *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 1004–1009, 2005, DOI: http://dx.doi.org/10.1109/TRO.2005.851383.

[WCH92] J. Weng, P. Coher, and M. Herniou, “Camera Calibration with Distortion Models and Accuracy Evaluation,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 10, pp. 965–980, 1992, DOI: http://dx.doi.org/10.1109/34.159901.

[WSZL08] J. Wang, F. Shi, J. Zhang, and Y. Liu, “A new calibration model of camera lens distortion,” in *Pattern Recognition*, vol. 41, no. 2, pp. 607–615, 2008, DOI: http://dx.doi.org/10.1016/j.patcog.2007.06.012.

[YH04] X. Ying and Z. Hu, “Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model,” in *Computer Vision - ECCV*, vol. 3021. Berlin, Germany: Springer Verlag, 2004, pp. 442–455, DOI: http://dx.doi.org/10.1007/978-3-540-24670-1_34.

[YM04] J. Yu and L. McMillan, “General linear cameras,” in *Computer Vision - ECCV*, vol. 3022. Berlin, Germany: Springer Verlag, 2004, pp. 14–27, DOI: http://dx.doi.org/10.1007/978-3-540-24671-8_2.

[Zha99] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” in *Proceedings - IEEE International Conference on Computer Vision*, vol. 1, Kerkyra, Greece, 1999, pp. 666–673, DOI: http://dx.doi.org/10.1109/iccv.1999.791289.

[Zha00] Z. Zhang, “A flexible new technique for camera calibration,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000, DOI: http://dx.doi.org/10.1109/34.888718.

[ZSJ+17] H. Zhao, S. Shi, H. Jiang, Y. Zhang, and Z. Xu, “Calibration of AOTF-based 3D measurement system using multiplane model based on phase fringe and BP neural network,” in *Optics Express*, vol. 25, no. 9, pp. 10 413–10 433, 2017, DOI: http://dx.doi.org/10.1364/oe.25.010413.

[ZWY11] H. Zhu, X. Wang, and C. Yi, “An elliptical function model for fisheye camera correction,” in *Proceedings - IEEE World Congress on Intelligent Control and Automation*, Taipei, Taiwan, 2011, pp. 248–253, DOI: http://dx.doi.org/10.1109/WCICA.2011.5970737.

[ZZH13] Y. Zhang, L. Zhao, and W. Hu, “A Survey of Catadioptric Omnidirectional Camera Calibration,” in *International Journal of Information Technology and Computer Science*, vol. 5, no. 3, pp. 13–20, 2013, DOI: http://dx.doi.org/10.5815/ijitcs.2013.03.02.

Publications by the Author

[1] Johannes Beck and Christoph Stiller. Non-parametric lane estimation in urban environments. In *Proceedings - IEEE Intelligent Vehicles Symposium*, pages 43–48, Dearborn, MI, USA, 2014. DOI: http://dx.doi.org/10.1109/IVS.2014.6856551.

[2] Johannes Beck and Christoph Stiller. Generalized b-spline camera model. In *Proceedings - IEEE Intelligent Vehicles Symposium*, pages 2137–2142, Changshu, China, 2018. DOI: http://dx.doi.org/10.1109/IVS.2018.8500466.

[3] Hannes Harms, Johannes Beck, Julius Ziegler, and Christoph Stiller. Accuracy analysis of surface normal reconstruction in stereo vision. In *Proceedings - IEEE Intelligent Vehicles Symposium*, pages 730–736, Dearborn, MI, USA, 2014. DOI: http://dx.doi.org/10.1109/IVS.2014.6856436.

[4] Haohao Hu, Johannes Beck, Martin Lauer, and Christoph Stiller. Continuous Fusion of IMU and Pose Data Using Uniform B-Spline. In *Proceedings - IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 173–178, Karlsruhe, Germany, 9 2020. DOI: http://dx.doi.org/10.1109/MFI49285.2020.9235248.

[5] Henning Lategahn, Johannes Beck, Bernd Kitt, and Christoph Stiller. How to learn an illumination robust image feature for place recognition. In *Proceedings - IEEE Intelligent Vehicles Symposium*, pages 285–291, Gold Coast, QLD, Australia, 2013. DOI: http://dx.doi.org/10.1109/IVS.2013.6629483.

[6] Henning Lategahn, Johannes Beck, and Christoph Stiller. DIRD is an illumination robust descriptor. In *Proceedings - IEEE Intelligent Vehicles Symposium*, pages 756–761, Dearborn, MI, USA, 2014. DOI: http://dx.doi.org/10.1109/IVS.2014.6856421.

[7] Sven Richter, Johannes Beck, Sascha Wirges, and Christoph Stiller. Semantic Evidential Grid Mapping Based on Stereo Vision. In *Proceedings - IEEE International Conference on Multisensor Fusion and Integration for*

Intelligent Systems (MFI), pages 179–184, Karlsruhe, Germany, 9 2020. DOI: http://dx.doi.org/10.1109/MFI49285.2020.9235217.

[8] Tobias Strauß, Julius Ziegler, and Johannes Beck. Calibrating multiple cameras with non-overlapping views using coded checkerboard targets. In *Proceedings - IEEE International Conference on Intelligent Transportation Systems*, pages 2623–2628, Qingdao, China, 2014. DOI: http://dx.doi.org/10.1109/ITSC.2014.6958110.