Intrinsic camera calibration for 3D machine vision – accuracy estimation and learned self-calibration

Zur Erlangung des akademischen Grades einer

Doktorin der Ingenieurwissenschaften (Dr.-Ing.)

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

angenommene

Dissertation

von

M.Sc. Annika Hagemann

Tag der mündlichen Prüfung:	08.07.2024
Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr.-Ing. Markus Ulrich

Abstract

An accurate model of a camera's projection is essential for computer vision tasks ranging from visual SLAM to stereo vision and neural radiance fields. It builds the foundation for visual perception in automated driving and robotics, and it turns a camera into a measurement instrument.

Projection models are inferred during calibration, where prior research has proposed a variety of model formulations and calibration techniques. Yet, two aspects limit existing techniques: (i) inaccuracies in the inferred model are challenging to identify, as they are often superimposed by measurement noise, or unobservable because of geometric ambiguities in the data, and (ii) existing techniques typically require images of a known target, limiting their practical applicability.

To address these challenges, this thesis introduces two building blocks: First, an accuracy-aware approach to target-based calibration is proposed, which infers bias and uncertainty along with the projection model. Subsequently, a deep learning based approach to intrinsic camera self-calibration is introduced, which enables estimating a camera's projection model without targets, based on image sequences of an unknown environment.

To detect bias, residual calibration errors are decomposed into measurement noise and bias. This enables detecting false model assumptions and selecting adequate models. To estimate uncertainty, an approximated bootstrap approach is introduced, giving a non-parametric alternative to the existing parametric covariance estimator. Thereby, uncertainty can be estimated for real-world calibrations, without ideal assumptions on the distribution of measurement noise.

Finally, to infer a camera's projection model based on images of an unknown environment, a learned self-calibration approach is proposed. By integrating classical self-calibrating bundle adjustment into a deep learning model, the capabilities of deep neural networks for feature extraction and matching are leveraged, while multi-view constraints are explicitly imposed.

Experiments show that the presented techniques enable reliable accuracy estimation and more accurate self-calibration than existing approaches. Thereby, they can contribute to the accuracy and safety of camera-based 3D perception.

Kurzfassung

Um Kamerabilder für Techniken wie visuelle Lokalisierung, 3D-Rekonstruktion oder visuelle Odometrie nutzen zu können, ist ein genaues Projektionsmodell der Kamera erforderlich. Ein solches Projektionsmodell ist eine Grundvoraussetzung für visuelle Umfeldwahrnehmung, z.B. in der Robotik und beim automatisierten Fahren.

Projektionsmodelle werden im Rahmen der Kalibrierung bestimmt, wobei die bisherige Forschung verschiedene Modelle und Techniken entwickelt hat. Diese sind jedoch durch zwei Faktoren limitiert: (i) Abweichungen des geschätzten Modells von der realen Kamera sind nicht trivial zu erkennen, da sie häufig durch Messrauschen überlagert sind, oder aufgrund geometrischer Mehrdeutigkeiten in den Daten nicht beobachtet werden können. (ii) Kalibrierung basiert meist auf Bildern von bekannten Kalibriertargets, was ihre praktische Anwendbarkeit einschränkt.

Diese Einschränkungen werden in der vorliegenden Arbeit addressiert. Im ersten Teil werden Methoden entwickelt, welche eine Schätzung der systematischen Abweichungen und der Unsicherheit eines Projektionsmodells ermöglichen. Dabei wird zur Erkennung systematischer Abweichungen eine Methode entwickelt, welche Rückprojektionsfehler in Messrauschen und systematischen Fehleranteil zerlegt. Zur Schätzung der Unsicherheit wird ein nichtparametrisches approximiertes Bootstrap Verfahren vorgeschlagen, welches robust gegen Abweichungen von idealen, normalverteilten Daten ist.

Im zweiten Teil wird ein neuer Ansatz zur targetlosen Selbstkalibrierung entwickelt, welcher die Schätzung des Projektionsmodells einer Kamera basierend auf Bildsequenzen einer unbekannten Umgebung ermöglicht. Dabei wird ein tiefes neuronales Netz zur Merkmalsextraktion und Korrespondenzsuche mit einer klassischen nicht-linearen Least-Squares-Optimierung der Kameraparameter kombiniert.

Experimentelle Ergebnisse zeigen, dass die vorgestellten Techniken eine verlässliche Genauigkeitsschätzung ermöglichen, und dass der entwickelte Ansatz zur Selbstkalibrierung eine höhere Genauigkeit als existierende klassische und auf Deep Learning basierende Verfahren erzielt. Dadurch kann diese Arbeit zur Genauigkeit und Sicherheit von kamerabasierter Umfeldwahrnehmung beitragen.

Acknowledgements

This thesis was written during my doctoral studies at Bosch Research and the Karlsruhe Institute of Technology (KIT).

I would like to express my sincere gratitude to everyone who supported me and contributed to this work.

First, I would like to thank Prof. Dr.-Ing. Christoph Stiller for the academic supervision, his continuous support, and for the research environment at the KIT Institute of Measurement and Control Systems. I also want to thank Prof. Dr.-Ing. Markus Ulrich for taking on the role of co-examiner, and Prof. Dr.-Ing. Arne Roennau for taking on the role of examination chair.

I am especially grateful to Dr.-Ing. Moritz Knorr, who inspired my interest in computer vision and advised this thesis at Bosch Research. His experience in topics like geometry, optimization, and calibration has been invaluable and I am very thankful for the many insightful discussions and the enthusiasm with which he advised this thesis.

I also want to thank my colleagues at Bosch Research, who have made my daily work an enriching experience. In particular, I would like to mention Jan Fabian Schmid, Charlotte Arndt, Christian Homeyer, and Holger Janssen, with whom I have shared a large part of the journey, and whom I would like to thank for numerous discussions, proof-readings and helpful feedback.

I am also thankful to my colleagues at the KIT Institute of Measurement and Control Systems for their constant support and for many long and insightful discussions and feedback. On behalf of everyone, I would like to mention Annika Meyer, Franziska Henze and Jan-Hendrik Pauls. In particular, the summer seminars will remain very positive memories for me, and I am glad to have been part of this institute.

Finally, I would like to thank my friends, my family, and my partner, who have supported me in so many ways.

Contents

Appendix

Notation and symbols

General notation

Scalars	Italic lowercase letters	a, α
Vectors	Bold lowercase letters	$\mathbf{a}$
Matrices	Bold uppercase letters	$\mathbf{A}$
Sets	Calligraphic uppercase letters	$\mathcal{A}$
Measurements	Variable letter with asterisk	α^*
Estimates	Variable letter with hat	$\widehat{\alpha}$
Ground-truth values	Variable letter with bar	$\bar{\alpha}$

Mathematical symbols

$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathbb{R}^+$	Positive real numbers
$SE(3)$	Special Euclidean group in three dimensions
$SO(3)$	Special Orthogonal group in three dimensions
$\mathbb{H}$	Quaternion algebra
$\mathcal{N}$	Gaussian normal distribution
χ_n^2	n-dimensional χ^2 distribution
$\mathbb{1}$	Identity matrix
$\mathbb{E}(\cdot)$	Expected value

$\|\| \cdot \|\|, \|\| \cdot \|\|_2$	Euclidean norm (L_2 norm)
$\|\| \cdot \|\|_1$	L_1 norm
$\|\| \cdot \|\|_{\boldsymbol{\Sigma}}$	Mahalanobis norm with $\|\|\mathbf{x}\|\|_{\boldsymbol{\Sigma}} = \sqrt{\mathbf{x}^\top \boldsymbol{\Sigma}^{-1} \mathbf{x}}$
$\otimes$	Convolution operation
$\odot$	Element-wise multiplication

Optimization and estimation

$\boldsymbol{\beta}$	Model parameters
ϵ	Measurement noise
σ^2	Measurement or detector variance
n	Number of observations
d	Number of parameters
$E(\cdot)$	Cost function
$\mathbf{r}$	Residuals
$\mathbf{J}$	Jacobian
$\mathbf{H}$	Approximated Hessian
$\mathbf{W}$	Weight matrix
λ	Damping factor (Levenberg Marquardt)
$\boldsymbol{\Sigma}$	Covariance matrix

Deep neural networks

$\mathcal{N}$	Neural network
$\mathbf{w}$	Weights of a neural network
$\mathbf{h}$	Hidden state of a recurrent neural network
$\mathcal{F}$	Feature map
$\mathcal{L}(\cdot)$	Loss function

3D geometry

$\mathbf{x}$	Point in 3D space
x, y, z	Coordinates of 3D point
$\mathbf{G}$	Pose in SE(3)
$\mathbf{R}$	Rotation matrix in SO(3)
$\mathbf{t}$	Translation vector
$\mathbf{G} * \mathbf{x}$	Pose transformation $\mathbf{G} * \mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{t}$

Projection and image space

$\mathbf{I}$	Image
$\mathbf{u}$	Point in 2D image space
u, v	Image coordinates
$\pi_C(\cdot)$	Camera projection function
$\mathbf{z}$	Pixel-wise depth
θ	Intrinsic camera parameters
f_x, f_y	Focal lengths in x- and y-direction
c_x, c_y	Principal point
$\mathbf{K}$	Camera matrix
$\kappa_1, \kappa_2, \kappa_3$	Radial distortion parameters
ξ	Parameter of the unified camera model
W, H	Image width and height
$\mathbf{f}_{ij}$	Optical flow between images $\mathbf{I}_i$ and $\mathbf{I}_j$

Indexing

k,t	Indexing of iterations
i, j	Indexing of images
ℓ	Indexing of 3D points
ι	Indexing of measurements
m, l	Indexing of parameters

Acronyms

MSE	Mean squared residual error
RMSE	Root mean squared residual error
MAD	Median absolute deviation
BR	Bias ratio
ME	Effective mapping error
EME	Expected mapping error
ATE	Average trajectory error
NLLS	Non-linear least squares
SfM	Structure-from-Motion
SLAM	Simultaneous localization and mapping
BS	Bootstrap method
aBS	Approximated bootstrap method
LiDAR	Light Detection and Ranging
CNN	Convolutional neural network
GRU	Gated recurrent unit
convGRU	Convolutional gated recurrent unit
$\mathrm{SC} - \mathrm{BA}$	Self-calibrating bundle adjustment
GPU	Graphics processing unit

NeRF	Neural Radiance Field

1 Introduction

Vision is not just a core element of human perception, but also an increasingly important component of technological systems, realized by cameras. An illustrative example is NASA's Mars Perseverance Rover (Fig. 1.1), a robotic system designed for exploring the surface of Mars [Far20]. It is equipped with a calibrated camera system, enabling it to measure the 3D structure of the environment, and providing high-resolution images of the surface of Mars [Mak20].

Besides large research projects that use camera systems as measurement instruments [Far20, Mic22], technologies such as automated driving [Mar19, Hän17] and robotics [Rub19] rely on camera-based environment perception, where applications range from industry to medical surgery [Özg20, Bar09]. Compared to complementary sensor modalities, such as LiDAR and radar, cameras have the advantage of being a low-cost sensor that provides rich information at high spatial as well as temporal resolution [Mar19].

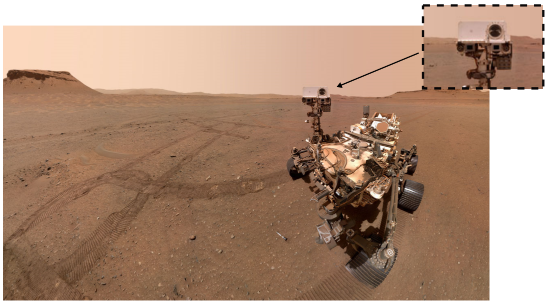


Figure 1.1: Mars Perseverance Rover equipped with a calibrated camera system on Jan. 22, 2023. Image obtained through image stitching, using a set of images taken by a camera on the rover's robotic arm. The close-up shows MastCam-Z, one of the main research camera systems. Image source and credits: NASA/JPL-Caltech [NAS23].

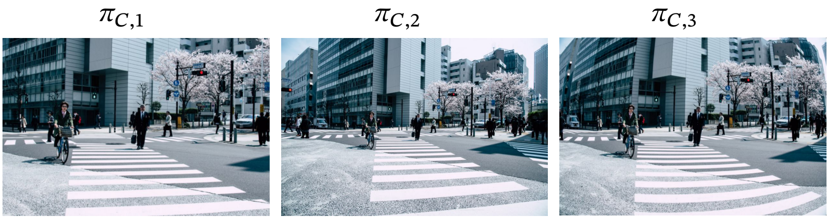


Figure 1.2: Different projection functions will result in different images of the same scene. To extract information on the three-dimensional geometry of the environment, the projection function π_C must be known which is achieved through calibration. Image of a street scene in Japan, synthetically transformed to show the impact of different projection models. Original image source: [Pxh18].

Besides information on color and visual appearance, camera images can provide information on the three-dimensional geometry of the environment. For instance, the stereo camera system of the Mars Rover allows it to measure the distance and the exact size of objects in its surroundings [Mak20]. Likewise, camera systems of automated vehicles enable measuring the locations of other traffic participants which is crucial to navigate safely [Mar19]. To extract such geometric information from images, the camera's mapping of the three-dimensional world to the two-dimensional image must be precisely known, in other words, the camera must be calibrated (see Fig. 1.2).

Camera calibration determines a mathematical function $\pi_C : \mathbb{R}^3 \to \mathbb{R}^2$ that maps every point in the 3D world to the image coordinates onto which it is projected by the camera [Zha00, Tsa87]. This projection function provides the bridge between the image and the geometry of the 3D world, and is a prerequisite for techniques such as visual SLAM [Mur15], stereo vision [Pen22], visual localization [Sar19] and neural radiance fields (NeRFs) [Tan22].

As a camera's projection is sensitive to the smallest variabilities in the lens system, π_C usually cannot be inferred from the building blocks of a camera alone but has to be inferred indirectly from images taken by the camera (Fig. 1.3).

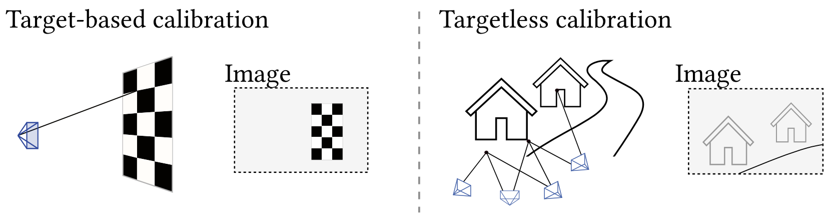


Figure 1.3: Target-based and targetless camera calibration. In target-based calibration, targets of well-known geometry are imaged. The image coordinates of points on the target are then used to infer the camera's projection function. Targetless calibration relies on images of an unknown environment.

In **target-based calibration**, this is achieved by taking images of well-known 3D objects, called calibration targets [Tsa87, Zha00]. The basic idea has already been used in early photogrammetry [Dua71] and since the seminal work by Zhang et al. [Zha00], which proposed to use planar calibration targets (Fig. 1.3), it has become accessible with relatively little equipment.

In **targetless calibration**, also referred to as self-calibration, π_C is inferred based on images of an unknown environment. This can be achieved by exploiting the consistency of the 3D structure across multiple images [Fau92, May92, Sch16a], or by relying on assumptions on the 3D structure, e.g. the presence of straight lines or right angles [Cip99, Ant17].

In the past years, many advances have been made in terms of calibration accuracy [Bec18, Sch20], as well as applicability [Pen19] of calibration techniques. Yet, achieving accurate calibration remains to be a challenge in the development of 3D computer vision systems [Hen15a, Cvi22, Cvi21]. In particular, two aspects limit existing approaches to camera calibration:

Model assessment Any model π_C inferred during calibration can be subject to biases and uncertainties and thus deviate from the camera's true physical projection. However, detecting such deviations is non-trivial, as they can be hidden behind measurement noise (cf. Chap. 3), or unobservable because of geometric ambiguities in the data (cf. Chap. 4). As a consequence, a defective calibration can easily be classified as sufficiently accurate and deployed in a system. This can diminish a system's overall performance, as recently showcased for two widely used computer vision datasets [Cvi22, Cvi21].

Reliable self-calibration Camera self-calibration could not only avoid the laborious target-based calibration process, it could also enable re-calibration during application, enhancing the safety of systems that rely on calibrated cameras [Hän17]. However, camera self-calibration is challenging because it cannot rely on prior knowledge on the environment. Although there exists a variety of approaches to self-calibration [Fau92, May92, Sch16a, Cip99, Ant17, Fan22, Gor19], the associated challenges have prevented it from replacing target-based calibration as the standard[1], despite its potential advantages.

The goal of this thesis is to address these challenges, enabling accurate intrinsic calibration, both, target-based and targetless. Thereby, this thesis should contribute to the accuracy and safety of systems that rely on camera-based 3D perception.

1.1 Problem statement

Any deviation between the inferred model π_C and the camera's true projection can impact downstream functions, including 3D-reconstruction, stereo vision and visual odometry [Ozo13, Svo96, Che04, Zuc01, Che11, Abr98]. If undetected, this can diminish a system's overall performance and pose a risk in safety-critical applications, such as automated driving [Hän17]. However, calibration inaccuracies are difficult to detect and existing metrics are limited (cf. Chap. 3,4), so that it usually requires expert knowledge to ensure accurate calibration [Cvi22, Cvi21]. Therefore, the first goal of this thesis is to introduce mathematical tools to reliably assess the quality of a calibration, giving an *accuracy-aware* approach to target-based calibration.

[1] As suggested by the number of scientific publications (e.g. [Pen19, Sch20, Cae20, Mak20]), computer vision software (e.g. [ROS21, Ope23]) and patents (e.g. [IRR23, GOS21, WAK20, LIU20, LU21, Val17]) that rely on target-based calibration.

In self-calibration, the main challenge is that neither the 3D structure, nor the motion of the camera can be controlled. This makes it difficult to reliably extract constraints on the projection model. Existing approaches to self-calibration either rely on assumptions on the 3D structure (e.g. the presence of straight lines or right angles) [Cip99, Ant17], or they use multiple images of the same scene and exploit the consistency of the 3D structure across images [Sch16a] (Fig. 1.3). While the former will be constrained to environments that fulfill the underlying assumptions, the latter can function in a vast range of environments, but its realization is challenging: It requires not only diverse camera perspectives to render all parameters observable [Stu97], but also accurate correspondences across images which is particularly challenging in the presence of moving objects, occlusions, little structure or repetitive structures [Sar20, Sun21, Sap18]. Therefore, the second goal of this thesis is to build upon recent advances in solving geometric computer vision tasks through deep learning (e.g. [Tee20, Lin21, Sar21, Lag20]), to make a step towards *accurate targetless camera calibration.*

1.2 Contributions

General remark The main contributions of this thesis have been prepublished in the following papers [Hag23, Hag22c, Hag22b, Hag20c] with associated patent applications [Hag20a, Hag20b, Hag21a, Hag21b, Hag22a].

The contributions of this thesis can be divided into four parts (Fig. 1.4). The first three parts focus on target-based calibration, introducing an accuracy-aware approach to target-based calibration. The fourth part moves to targetless calibration and introduces a novel approach to intrinsic self-calibration from video. Overall, the contributions are the following:

1 A **novel approach to detect calibration biases** [Hag20c, Hag22b], called the *bias ratio*, that disentangles biases in the inferred projection model from measurement noise. Thereby, even small inaccuracies in the calibration setup can be detected.

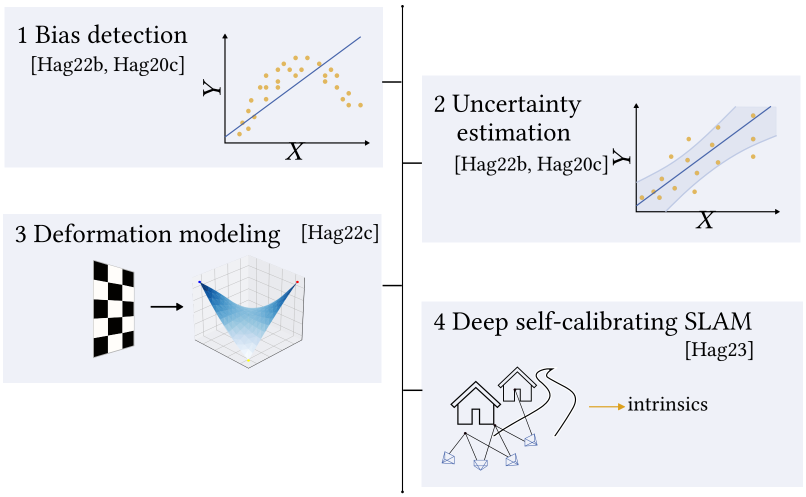


Figure 1.4: Overview of the main contributions of this thesis.

2 A **non-parametric approach to infer calibration uncertainty** [Hag20c, Hag22b], covering:

- Experimental evidence that the commonly used parametric approach to uncertainty estimation underestimates calibration uncertainty on real data.
- A novel **approximated bootstrap approach** that estimates the covariance matrix of model parameters without parametric assumptions.
- A **model-independent uncertainty metric**, called the *expected mapping error*, that expresses the calibration uncertainty in image space, in terms of a model-independent scalar value.

3 Experimental evidence that manually handling a calibration target can lead to small flexible target deformations that introduce significant bias into a calibration, and a **deformable target model** which enables accurate calibration despite moving the target during data acquisition [Hag22c].

4 A **learned approach to camera self-calibration from monocular video** [Hag23] that combines deep learning with explicit modeling of projection functions and multi-view geometry. This includes:
   - A model consisting of a deep neural network and a **self-calibrating bundle adjustment (SC-BA) layer**, which enables estimating intrinsic camera parameters from video. While the neural network is trained to predict weighted correspondences between pairs of images with overlapping field-of-view, the SC-BA layer optimizes the intrinsics through differentiable bundle adjustment.
   - A realization of the approach within a deep visual SLAM system [Tee21], giving a **self-calibrating visual SLAM system** that infers camera motion, depth and intrinsics from monocular video.

All contributions are evaluated experimentally and compared with the state-of-the-art approaches in their respective field. In combination, they enable quantifying and ensuring the accuracy of target-based calibration and provide a step towards accurate intrinsic self-calibration.

1.3 Outline

After introducing the fundamentals of estimation theory and 3D computer vision in Chap. 2, the thesis is structured such that each of the Chap. 3-6 contains one of the contributions (see Sec. 1.2). In line with this structure, each chapter contains a separate section on the state-of-the-art of the addressed field, replacing an all-encompassing state-of-the-art chapter.

In **Chap. 2** the fundamentals are introduced. This chapter builds the foundation of all subsequent chapters, as it introduces the relevant terms and the nomenclature that is used throughout the thesis.

Chap. 3 addresses the first contribution, namely the detection of biases in target-based calibration. It introduces the *bias ratio* [Hag22b, Hag20c], which quantifies the fraction of bias in the residual errors of a calibration. The method is assessed experimentally on multiple datasets and compared with existing approaches to detect biases.

Chap. 4 addresses the second contribution, namely uncertainty estimation in camera calibration. In Sec. 4.2.2, it is shown that real calibrations deviate from the ideal assumptions underlying the standard approach to uncertainty estimation in camera calibration resulting in a significant underestimation of the uncertainty. To overcome this issue, an *approximated bootstrap approach* [Hag22b] to uncertainty estimation is proposed which is non-parametric and thus robust to deviations from ideal assumptions. Sec. 4.3 then introduces an uncertainty metric, the *expected mapping error* [Hag22b, Hag20c], that quantifies uncertainty in a model-independent manner.

Chap. 5 moves from the *detection* of inaccuracies to the *prevention.* Based on the metrics introduced in Chap. 3-4, it is shown that moving a calibration target during data acquisition can lead to small flexible target deformations that can introduce significant biases into a calibration [Hag22c]. To prevent such biases, a novel approach to modeling flexible target deformations [Hag22c] is introduced. The improvement in calibration accuracy is demonstrated on multiple datasets and in a Structure-from-Motion application.

Chap. 6 addresses targetless calibration, also referred to as self-calibration. After reviewing the state-of-the-art on intrinsic camera self-calibration, a novel approach is introduced, which combines deep learning with explicit modeling of projection functions and multi-view geometry [Hag23]. The calibration accuracy is evaluated on multiple public datasets and compared to state-of-the-art methods.

General remark For the sake of readability, the first person plural will be used throughout this thesis.

2 Fundamentals

This chapter introduces the fundamentals that the thesis builds upon, including camera projection models (Sec. 2.1), the mathematical basis of estimation theory and optimization (Sec. 2.2), and the employed target-based calibration formulation (Sec. 2.3). Furthermore, fundamentals of multi-view computer vision are introduced, which are required for self-calibration (Sec. 2.4).

2.1 Projection models

The formation of an image in a camera involves a combination of physical phenomena, ranging from the refraction of light when passing the lens system [För16, p.477], to the photoelectric effect in the image sensor [Mag03]. Despite this complexity, the *geometry* of image formation can be approximated by comparatively simple mathematical models.

We define the *camera coordinate system* with coordinates $\mathbf{x} = (x, y, z)^\top \in \mathbb{R}^3$ to originate in the camera's optical center[1], with the z-axis coinciding with the optical axis and pointing into viewing direction (Fig. 2.1). The *image coordinate system* is a two-dimensional coordinate system with coordinates $\mathbf{u} = (u,v)^\top \in \mathbb{R}^2$ that originates in the upper left corner of the image. The camera's projection from the 3D world to the 2D image can now be expressed as a function $\pi_C : \mathbb{R}^3 \to \mathbb{R}^2, \mathbf{x} \mapsto \mathbf{u}$, referred to as the *projection model* or *camera model*. Its model parameters are called the intrinsic camera parameters, or *intrinsics* $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, $n_\theta \in \mathbb{N}$. The intrinsics $\boldsymbol{\theta}$ characterize a camera's

[1] Throughout this thesis, we use central camera models, assuming a single optical center.

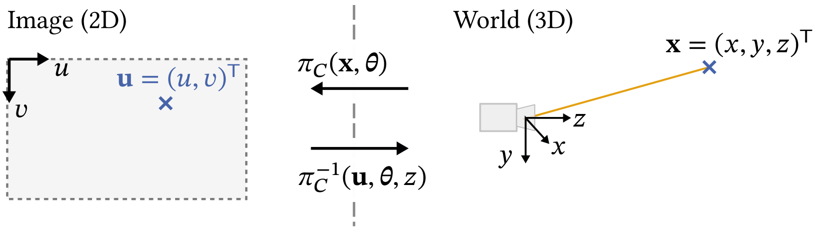


Image (2D)
u
v
$\mathbf{u} = (u, v)^\top$
$\pi_C(\mathbf{x}, \theta)$
$\pi_C^{-1}(\mathbf{u}, \theta, z)$
World (3D)
$\mathbf{x} = (x, y, z)^\top$
z
x
y

Figure 2.1: Camera projection from the three-dimensional world to the two-dimensional image. Two-sided depiction inspired by [Bec21, Fig. 4.1].

projection by quantifying possible lens distortions, the camera's focal length, and the dimension of the image sensor.

The backward projection maps an image point to its associated viewing ray. Recovering a single 3D point requires knowledge about the point's depth $z \in \mathbb{R}$, i.e. the z-coordinate of the 3D point in the camera coordinate system. We define $\pi_C^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3, (\mathbf{u}, z) \mapsto \mathbf{x}$ to denote the mapping of an image point to a 3D world point, given the depth z of the point. To show the dependence of π_C and π_C^{-1} on the intrinsics θ, they will be written as $\pi_C(\mathbf{x}, \theta)$ and $\pi_C^{-1}(\mathbf{u}, \theta, z)$ in the following.

In general, the techniques presented throughout this thesis are independent of the specific choice of the projection model. In the experiments, we will focus on four commonly used models, which will be introduced in the following.

Pinhole model

The pinhole model [Har04, p.154-155] describes a central projection without distortion. The projection π_C and inverse projection π_C^{-1} are defined as follows [Hag23]:

$$\pi_C(\mathbf{x}, \theta) = \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix} \quad \text{and} \quad \pi_C^{-1}(\mathbf{u}, \theta, z) = z \begin{bmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ 1 \end{bmatrix}, \tag{2.1}$$

where the intrinsics are given by $\theta = (f_x, f_y, c_x, c_y)$, containing the focal lengths f_x, $f_y \in \mathbb{R}$ and the principal point $(c_x, c_y) \in \mathbb{R}^2$. The pinhole model can also be written in terms of the *camera matrix* $\mathbf{K} \in \mathbb{R}^{3\times3}$ [Har04, p.155].

Radial distortion model

Introducing polynomial terms allows modeling lens distortion [Bro66]. In the following, a model with three radial distortion parameters κ_1, κ_2, $\kappa_3 \in \mathbb{R}$ is used,

$$\pi_C(\mathbf{x}, \theta) = \begin{bmatrix} f_x \frac{x}{z}(1 + \kappa_1\rho^2 + \kappa_2\rho^4 + \kappa_3\rho^6) + c_x \\ f_y \frac{y}{z}(1 + \kappa_1\rho^2 + \kappa_2\rho^4 + \kappa_3\rho^6) + c_y \end{bmatrix}, \tag{2.2}$$

where $\rho = \sqrt{\left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2}$ is the distance from the optical axis in a normalized image plane.

The polynomial distortion model $d(\rho) = \rho(1 + \kappa_1\rho^2 + \kappa_2\rho^4 + \kappa_3\rho^6)$ cannot be inverted analytically. However, it can be inverted numerically for each given image point $\mathbf{u}$ (see App. Sec. A.5).

Fisheye model

Fisheye lenses are wide-angled lenses, which introduce distortions that can result in a field of view beyond 180°. To model such lenses, we use the *opencv* fisheye model [Ope21], which is defined by the following projection function[1]:

$$\pi_C(\mathbf{x}, \theta) = \begin{bmatrix} f_x \frac{x}{z}\frac{\gamma_d}{\rho} + c_x \\ f_y \frac{y}{z}\frac{\gamma_d}{\rho} + c_y \end{bmatrix}, \qquad \rho = \sqrt{\left(\frac{x}{z}\right)^2 + \left(\frac{y}{z}\right)^2}, \tag{2.3}$$

[1] Note that the formulation of the opencv fisheye model diverges at $z = 0$, i.e. at a field of view close to 180°. Despite this disadvantage, it is widely used [Ope21, Sch16a]. To resolve this divergence, the variant employed in e.g. [Ric13] can be used.

where $\gamma_d = \gamma(1+\kappa_1\gamma^2+\kappa_2\gamma^4+\kappa_3\gamma^6+\kappa_4\gamma^8)$, with $\gamma = \arctan(\rho)$ and fisheye distortion parameters $\kappa_1,\kappa_2,\kappa_3,\kappa_4 \in \mathbb{R}$.

Unified camera model

The unified camera model [Mei07] can approximate a wide range of lenses, from pinhole to fisheye and catadioptric optics. The projection is given by

$$\pi_C(\mathbf{x}, \boldsymbol{\theta}) = \begin{bmatrix} f_x \frac{x}{z+\xi\|\mathbf{x}\|} + c_x \\ f_y \frac{y}{z+\xi\|\mathbf{x}\|} + c_y \end{bmatrix}, \tag{2.4}$$

where $\|\mathbf{x}\| = \sqrt{x^2+y^2+z^2}$ and $\xi \in \mathbb{R}$. Similar to the pinhole model, the unified model can be inverted analytically. The inverse projection is given by

$$\pi_C^{-1}(\mathbf{u}, \boldsymbol{\theta}, z) = z \begin{bmatrix} \frac{\beta x_s}{(\beta-\xi)} \\ \frac{\beta y_s}{(\beta-\xi)} \\ 1 \end{bmatrix}, \qquad \beta = \frac{\xi + \sqrt{1+(1-\xi^2)(x_s^2+y_s^2)}}{1+x_s^2+y_s^2}, \tag{2.5}$$

where $x_s = \frac{u-c_x}{f_x}$ and $y_s = \frac{v-c_y}{f_y}$.

The different camera models described in this section can approximate the projections of a vast range of lenses. During calibration, a suitable model π_C is selected, and the associated intrinsics $\boldsymbol{\theta}$ are estimated based on measurements acquired with the camera. The following section introduces the fundamentals of model fitting and estimation that are the basis of calibration.

2.2 Model fitting and estimation

Camera calibration is a model fitting problem, as it aims at finding a model that accurately describes a camera's projection, based on measurements of

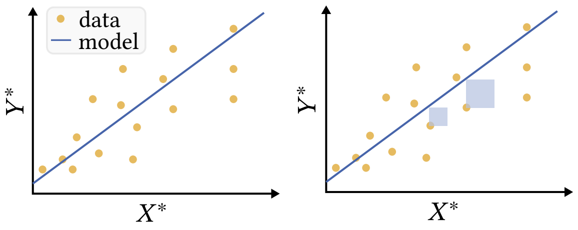


Figure 2.2: Simple 1D example of model fitting (left) and specifically least squares estimation (right) which serves to introduce the fundamental concepts of estimation theory that underlie camera calibration.

the camera. As such, it is conceptually similar to modeling the relation between two variables, X and Y, that is, a simple one-dimensional model fitting problem (Fig. 2.2).

2.2.1 Model fitting

Consider the task of identifying a model that best approximates the underlying relationship between two variables X and Y based on $n \in \mathbb{N}$ noisy measurements $(\mathbf{X}^*, \mathbf{Y}^*)$, where $\mathbf{X}^*, \mathbf{Y}^* \in \mathbb{R}^n$. Assume the measurements $\mathbf{X}^*$ to be exact, i.e. $\mathbf{X}^* = \mathbf{X}$, but the measurements $\mathbf{Y}^*$ to be subject to measurement noise, i.e. $\mathbf{Y}^* = \mathbf{Y} + \boldsymbol{\epsilon}$ with random errors[1] $\boldsymbol{\epsilon} \in \mathbb{R}^n$. Regardless of the dimension and meaning of these variables, such a model fitting problem can be addressed by

1 selecting a functional model $\hat{\mathbf{Y}} = \mathbf{f}(\mathbf{X}, \boldsymbol{\beta}) \in \mathbb{R}^n$ where $\boldsymbol{\beta} \in \mathbb{R}^d$ denotes the unknown model parameters,

2 defining a cost function $E(\boldsymbol{\beta}) \in \mathbb{R}$ that quantifies the deviation of model predictions $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta})$ from the measurements $\mathbf{Y}^*$,

[1] The term *error* may also be replaced with the more precise term *deviation*. However, in alignment with common practice in the literature, this thesis continues to use the term *error* in several places.

3 estimating the model parameters by optimizing the cost function $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} E(\boldsymbol{\beta})$.

The last step can be achieved through different estimation techniques. In the following, *non-linear least squares estimation* will be introduced, as the calibration problem, as well as many other optimization problems in computer vision, are formulated as non-linear least squares problems.

2.2.2 Least squares estimation

In general, a problem is called a least squares (LS) problem if its solution minimizes the sum of squared error terms (Fig. 2.2) [Noc06, p.245]:

$$\text{LS problem:} \quad E(\boldsymbol{\beta}) = \frac{1}{2}\sum_{\iota=1}^{n} r_\iota(\boldsymbol{\beta})^2 = \frac{1}{2}||\mathbf{r}(\boldsymbol{\beta})||^2 = \frac{1}{2}\mathbf{r}(\boldsymbol{\beta})^\top\mathbf{r}(\boldsymbol{\beta}), \tag{2.6}$$

where the error terms $r_\iota(\boldsymbol{\beta}) : \mathbb{R}^d \mapsto \mathbb{R}$ are functions of the model parameters $\boldsymbol{\beta}$, also referred to as the *residual errors* or *residuals*, with $\iota = 1,...n$. They can be combined to form the *residual vector* $\mathbf{r}(\boldsymbol{\beta}) = (r_1(\boldsymbol{\beta}),...r_n(\boldsymbol{\beta}))^\top : \mathbb{R}^d \mapsto \mathbb{R}^n$. In model fitting, the residuals are given by the difference between model predictions $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta})$ and observations $\mathbf{Y}^*$:

$$\mathbf{r}(\boldsymbol{\beta}) = \mathbf{f}(\mathbf{X}, \boldsymbol{\beta}) - \mathbf{Y}^*. \tag{2.7}$$

Depending on the structure of the function $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta})$, one distinguishes *linear least squares* and *non-linear least squares estimation* (see App. Sec. A.2). In the following non-linear least squares estimation will be addressed in further detail, as it will be used throughout this thesis.

Non-linear least squares In *non-linear least squares (NLLS)* problems, the function $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta})$ contains non-linear dependencies on the parameters $\boldsymbol{\beta}$. Unlike linear least squares problems, which can be solved analytically (see App. Sec. A.2), non-linear least squares problems must be solved iteratively. For the sake of generality, we formulate the following equations for a *weighted* non-linear least squares problem which additionally introduces a weight matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ into the cost function (2.6)[1]:

$$\text{Weighted NLLS problem}: \quad E(\boldsymbol{\beta}) = \frac{1}{2}\mathbf{r}(\boldsymbol{\beta})^{\top}\mathbf{W}\mathbf{r}(\boldsymbol{\beta}). \tag{2.8}$$

Two of the most common algorithms to optimize non-linear least squares problems are the *Gauss-Newton algorithm* and the *Levenberg Marquardt algorithm* [Tri99]:

Gauss-Newton algorithm The idea behind the Gauss-Newton algorithm is to locally approximate the residuals with a linear function and to perform iterative steps towards convergence[2]. This local approximation gives rise to the *Gauss-Newton update* which is given by [Noc06, p.254]

$$\begin{aligned} \mathbf{J}^{\top}\mathbf{W}\mathbf{J}\boldsymbol{\Delta\beta} &= -\mathbf{J}^{\top}\mathbf{W}\mathbf{r}^{k} \\ \boldsymbol{\beta}^{k+1} &= \boldsymbol{\beta}^{k} + \boldsymbol{\Delta\beta}, \end{aligned} \tag{2.9}$$

where $k \in \mathbb{N}$ is the update iteration, $\boldsymbol{\Delta\beta} \in \mathbb{R}^{d}$ is the parameter update and $\mathbf{J} \in \mathbb{R}^{n\times d}$ is the *Jacobian* of the residuals w.r.t. the parameters $\boldsymbol{\beta} = (\beta_1,...\beta_d)^{\top}$, given by

$$\mathbf{J} = \left[\frac{\partial r_\iota}{\partial \beta_m}\right]_{\substack{\iota=1,...n \\ m=1,...d}}, \tag{2.10}$$

evaluated at the current parameter values $\boldsymbol{\beta}^{k}$. The matrix $\mathbf{H} = \mathbf{J}^{\top}\mathbf{W}\mathbf{J} \in \mathbb{R}^{d\times d}$ is called the *approximated Hessian* of the quadratic cost function [Tri99]. Gauss-Newton updates (2.9) are performed iteratively until a pre-defined termination criterion is met.

Levenberg Marquardt algorithm To improve convergence in case the approximated Hessian is ill-conditioned, the *Levenberg Marquardt algorithm* [Lev44, Mar63] augments the Gauss-Newton step with a damping term $\lambda\mathbb{1}$, $\lambda \in \mathbb{R}^{+}$ and identity matrix $\mathbb{1} \in \mathbb{R}^{d\times d}$, giving the update step [Noc06,

[1] The weight matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ is typically chosen to reflect the variance of individual measurements, by setting it equal to the inverse covariance matrix of measurement noise $\mathbf{W} = \boldsymbol{\Sigma}_{\epsilon\epsilon}{}^{-1}$.

[2] App. Sec. A.2, contains a derivation of the algorithm.

p.258]

$$\begin{aligned}(\mathbf{J}^\top \mathbf{W}\mathbf{J} + \lambda \mathbb{1})\Delta\boldsymbol{\beta} &= -\mathbf{J}^\top \mathbf{W}\mathbf{r}^k \\ \boldsymbol{\beta}^{k+1} &= \boldsymbol{\beta}^k + \Delta\boldsymbol{\beta}.\end{aligned} \tag{2.11}$$

The damping factor λ is adjusted in every iteration and balances the update step between the Gauss-Newton direction and the gradient descent direction [Mar63] [Noc06, p.258-262]. Similar to the Gauss-Newton algorithm, updates (2.11) are performed until a pre-defined termination criterion is met.

The model parameters obtained after convergence of either optimization algorithm are the *estimated parameters*, which are denoted by $\hat{\boldsymbol{\beta}}$ in the following.

2.2.3 Model assessment

The goal of model assessment is to quantify how accurately a model describes the data, and how accurately it is expected to describe future, unseen data. This section introduces the relevant concepts associated with assessing a model. Chapters 3 and 4 will build upon these concepts to derive an accuracy-aware approach to camera calibration.

Bias and uncertainty

Following [För16, p.116], estimation theory generally distinguishes between *accuracy*, *bias* and *precision*. Given a sample of measurements or estimates $\{\beta_\iota\}_{\iota=1}^n$ of a variable β, the bias is defined by "the deviation of the estimated mean from the true value" [För16, p.116]. The precision, on the hand, is defined by "the deviation of repeated trials from their estimated mean" [För16, p.116]. Last, the accuracy encompasses both and is defined as "the deviation of repeated trials from the true value" [För16, p.116].

While these definitions are clear when it comes to estimating or measuring a specific parameter $\beta \in \mathbb{R}$, they become more vague when it comes to model fitting. In particular, the problem of using an insufficiently complex model

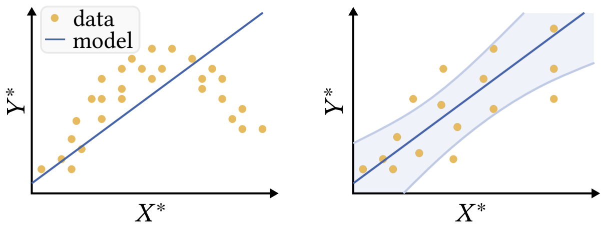


Figure 2.3: Bias and uncertainty in a 1D model fit.

(Fig. 2.3) is difficult to describe with the above definitions, as it results in different systematic deviations between model predictions $\hat{\mathbf{Y}}$ and true values $\mathbf{Y}$ at different positions $\mathbf{X}$ (Fig. 2.3). In the literature, this has resulted in the following concepts for assessing the quality of a model [Jam21, p.33-35] (see Fig. 2.3):

- **Bias**, also referred to as *underfit*, *systematic error* or *model-misspecification error*, describes deviations between model and data which are caused by a model not being sufficiently flexible to describe the data (Fig. 2.3, left).
- **Uncertainty**, also referred to as *variance*, describes how much the model is expected to fluctuate when repeatedly performing the estimation with different measurement samples (Fig. 2.3, right).

Assessing the quality of a model requires capturing both, bias and uncertainty. The quantities that are commonly evaluated for this purpose are the *residual errors* (2.7) and the *covariance matrix* (2.15) of the optimization.

Residual errors

Residual errors (2.7) are available after any optimization and provide information about how well the obtained model fits the training data, i.e. the dataset used during optimization. Commonly used measures are the *mean squared* *residual error* (MSE),

$$\mathrm{MSE} = \frac{1}{n}\mathbf{r}^\top \mathbf{r} = \frac{1}{n}\sum_{\iota=1}^{n} ||r_\iota||^2, \tag{2.12}$$

and the *root mean squared residual error* (RMSE),

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{\iota=1}^{n} ||r_\iota||^2}. \tag{2.13}$$

Large residual errors indicate that the model does not accurately describe the data. However, residual errors can be artificially reduced by using fewer measurements or more flexible models, e.g. with more model parameters. As such, residual errors are not able to capture the uncertainty of a model.

Covariance of model parameters

The uncertainty of a model can be quantified in terms of the variances and covariances of estimated model parameters $\hat{\boldsymbol{\beta}}$ [Har04, p.138-139]. They quantify how much the parameters are expected to fluctuate if estimation is performed using different data samples $\{(\mathbf{X}_1^*, \mathbf{Y}_1^*), ...(\mathbf{X}_{n_s}^*, \mathbf{Y}_{n_s}^*)\}$, $n_s \in \mathbb{N}$ that come from the same population as the original sample $(\mathbf{X}^*, \mathbf{Y}^*)$.

The *covariance matrix* $\boldsymbol{\Sigma}_{\hat{\beta}\hat{\beta}} \in \mathbb{R}^{d\times d}$ contains the variances σ_m^2 of each parameter β_m and the covariance σ_{ml} of all pairs of parameters:

$$\boldsymbol{\Sigma}_{\hat{\beta}\hat{\beta}} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{pmatrix}. \tag{2.14}$$

For linear least squares estimation, under the assumption that the measurement noise $\boldsymbol{\epsilon}$ is independently and identically distributed (i.i.d.), following a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\epsilon\epsilon})$ with covariance matrix $\boldsymbol{\Sigma}_{\epsilon\epsilon} = \operatorname{diag}(\sigma^2) \in \mathbb{R}^{n \times n}$ and variance $\sigma^2 \in \mathbb{R}$, it can be shown that the covariance matrix of model parameters is given by a backpropagation of the measurement noise [Har04, p.142]:

$$\widehat{\boldsymbol{\Sigma}}_{\hat{\beta}\hat{\beta}} = \sigma^2(\mathbf{J}^\mathsf{T}\mathbf{J})^{-1}, \tag{2.15}$$

where $\mathbf{J}$ is the Jacobian (2.10) of the residuals evaluated at $\hat{\boldsymbol{\beta}}$. As the variance of the measurement noise σ^2 is typically not known *a priori*, it can be approximated using the *estimated accuracy* [Luh13, p.92-96] [Har04, p.141-142],

$$\hat{s}^2 = \frac{\text{MSE}}{\left(1 - \frac{d}{n}\right)}, \tag{2.16}$$

where n is the number of observations and d the number of parameters. In non-linear least squares problems with i.i.d. Gaussian measurement noise, the same covariance estimator can be used as a first-order approximation for the covariance matrix [Har04, p.142].

2.3 Target-based camera calibration

Having introduced geometric projection models and estimation theory, this section introduces the target-based calibration formulation that is used through Chap. 3-5. This formulation is also used in [Hag22b, Hag20c, Hag22c]. We focus on monocular intrinsic calibration, but the techniques presented throughout this thesis are generally not limited to the monocular case.

Camera calibration determines the projection model $\pi_C(\mathbf{x}, \boldsymbol{\theta})$ that maps every point $\mathbf{x} \in \mathbb{R}^3$ in the 3D world to its associated image coordinates $\mathbf{u} \in \mathbb{R}^2$ (cf. Sec. 2.1). In target-based calibration, this is achieved by imaging an object of known geometry and appearance, the so-called calibration target. There exists a variety of approaches to target-based camera calibration [Tsa87, Zha00,

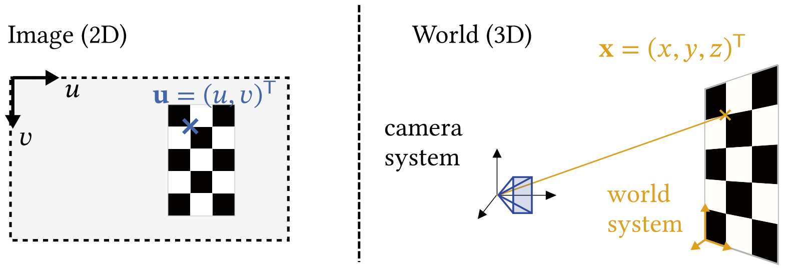


Figure 2.4: Target-based camera calibration using a checkerboard target.

Str14, Bec18, Sch20], the most prominent ones being Tsai's method [Tsa87] and Zhang's method [Zha00]. Throughout this thesis, we will build upon Zhang's method [Zha00], using a single planar checkerboard target (Fig. 2.4).

2.3.1 Representation of 3D poses

As the relative pose between target and camera center, also referred to as *extrinsics*, is not known a priori, it is estimated during calibration. A 3D pose $\mathbf{G}_i \in SE(3)$ is composed of a 3D rotation $\mathbf{R}_i \in SO(3)$ and a translation $\mathbf{t}_i \in \mathbb{R}^3$. The rotation $\mathbf{R}_i$ can be represented in different ways (see App. Sec. A.4), including a rotation matrix, a unit quaternion, and a three-component vector, whose direction defines the rotation axis while its length defines the rotation angle (axis-angle representation). During optimization, we use the axis-angle representation, as it avoids the overparametrization of other representations (see App. Sec. A.4). In the mathematical formulation throughout this thesis, we abstract from the choice of the representation for better readability and always use the symbol $\mathbf{G} \in SE(3)$ to denote 3D poses. We further define the symbol $*$ to denote the pose transformation of a 3D point in Cartesian coordinates $\mathbf{x} \in \mathbb{R}^3$, so that $\mathbf{G} * \mathbf{x} := \mathbf{R}\mathbf{x} + \mathbf{t}$ where $\mathbf{R} \in SO(3)$ denotes the rotation and $\mathbf{t} \in \mathbb{R}^3$ the translation of $\mathbf{G} \in SE(3)$.

2.3.2 Data acquisition and pre-processing

The calibration setup consists of a single monocular camera $\mathcal{C}$ and a checkerboard target $\mathcal{T}$ (Fig. 2.4). Both, the camera and the board define their own coordinate system: the camera coordinate system originates in the optical center of the camera, and the board coordinate system originates in one of the corners of the checkerboard (Fig. 2.4). As the target has been precisely manufactured or measured beforehand, its geometry is assumed to be known. Specifically, the 3D coordinates of the checkerboard corners $\mathcal{X}_\mathcal{B} = \{\mathbf{x}_b \in \mathbb{R}^3 | b = 1, ...N_{\mathcal{X}_\mathcal{B}}\}$ with respect to the board coordinate system (Fig. 2.4) are assumed to be known.

The calibration dataset is collected by taking a set of images $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{H \times W \times 3} | i = 1,...N_\mathcal{I}\}$ of the calibration target, using the camera $\mathcal{C}$. During data acquisition, camera and calibration target are moved with respect to each other, so that they are positioned at different relative distances and angles. This is required to render all model parameters observable [Pen19, Roj18].

After data collection, a corner detection algorithm [Str14] is used to extract the image coordinates $\mathcal{U}^* = \{\mathbf{u}^*_{ib} \in \mathbb{R}^2 | i = 1,...N_\mathcal{I},\ b = 1, ...N_{\mathcal{X}_\mathcal{B}}\}$ of projected checkerboard corners from the images. As corner detection is subject to measurement noise (also referred to as *detector noise*), the detected coordinates $\mathcal{U}^*$ will deviate from the true projections $\bar{\mathcal{U}}$ by an error term $\mathcal{E} = \{\epsilon_{ibl} \in \mathbb{R} | i = 1,...N_\mathcal{I},\ b = 1, ...,N_{\mathcal{X}_\mathcal{B}},\ l = 1,2\}$. The individual error terms are approximated to be i.i.d. Gaussian random variables so that the vector $\boldsymbol{\epsilon} \in \mathbb{R}^n, n = 2N_\mathcal{I}N_{\mathcal{X}_\mathcal{B}}$, that contains all error terms in $\mathcal{E}$ follows a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\epsilon\epsilon})$ with covariance matrix $\boldsymbol{\Sigma}_{\epsilon\epsilon} = \text{diag}(\sigma^2)$ and detector variance σ^2. The detected image coordinates $\mathcal{U}$ are associated with the known 3D coordinates $\mathcal{X}_\mathcal{B}$ by using coded targets [Str14] which resolves the association problem.

2.3.3 Estimation of model parameters

The detected image points $\mathcal{U}^*$ and associated 3D coordinates $\mathcal{X}_\mathcal{B}$ enable estimating the camera's intrinsics $\boldsymbol{\theta}$. To this end, the relation between a 3D point

$\mathbf{x}_b$ and its associated image point $\mathbf{u}_{ib}$ can be written as

$$\mathbf{u}_{ib} = \pi_C(\mathbf{G}_i * \mathbf{x}_b, \theta), \tag{2.17}$$

where π_C denotes the projection function (cf. Sec. 2.1) and $\mathbf{G}_i \in SE(3)$ denotes the 3D target pose, transforming the 3D point $\mathbf{x}_b$ from the target coordinate system to the camera coordinate system.

To generate an initial estimate of all poses $\mathbf{G} = (\mathbf{G}_1, ...\mathbf{G}_{N_\mathcal{I}})^\mathsf{T}$ we follow the approach described in [Str15], and the initial values for the intrinsics are obtained by setting the principal point to the image center, using the manufacturer information to obtain an initial guess for the focal length, and assuming a distortion-free camera.

Finally, given initial estimates for camera intrinsics and poses, they are refined during bundle adjustment by minimizing the *reprojection error* [Hag22b]:

$$\hat{\mathbf{G}}, \hat{\theta} = \underset{\mathbf{G},\theta}{\arg\min} \sum_{i=1}^{N_\mathcal{I}} \sum_{b=1}^{N_{\mathcal{X}_\mathcal{B}}} ||\mathbf{u}^*_{ib} - \pi_C(\mathbf{G}_i * \mathbf{x}_b, \theta)||^2. \tag{2.18}$$

This is a non-linear least squares problem (Sec. 2.2.2) which is solved iteratively using the Levenberg Marquardt algorithm (Sec. 2.2.2). To reduce the impact of potential spurious outliers, the cost function can be robustified, for instance, using a Cauchy kernel [Har04, p.616-619].

2.3.4 Modeling static target deformation

The calibration approach described above assumes the 3D coordinates of checkerboard corners $\mathcal{X}_\mathcal{B}$ to be known. However, calibration targets may be imperfect and thus deviate from the assumed geometry. To deal with imperfect calibration targets, the assumed 3D coordinates can be corrected during calibration [Str11, Str14]. To this end, an additional set of parameters $\mathbf{\Delta x} = (\mathbf{\Delta x}_1, ...\mathbf{\Delta x}_{N_{\mathcal{X}_\mathcal{B}}})^\mathsf{T}$ with $\mathbf{\Delta x}_b = (\Delta x, \Delta y, \Delta z)^\mathsf{T}$, can be introduced which describes the deviation of the 3D coordinates from the assumed geometry.

The cost function for bundle adjustment is then given by [Hag22c]:

$$\widehat{\mathbf{G}}, \hat{\theta}, \widehat{\mathbf{\Delta x}} = \underset{\mathbf{G},\theta,\mathbf{\Delta x}}{\arg\min} \sum_{i=1}^{N_{\mathcal{I}}} \sum_{b=1}^{N_{\mathcal{X}_{\mathcal{B}}}} ||\mathbf{u}_{ib}^{*} - \pi_C(\mathbf{G}_i * (\mathbf{x}_b + \mathbf{\Delta x}_b), \theta)||^2. \tag{2.19}$$

In the following, this formulation is referred to as *static deformation modeling*, as it allows for a target deformation that is fixed throughout the dataset. To resolve ambiguities between correction terms $\mathbf{\Delta x}$ and target poses, and to fix the overall scale of the target, seven degrees of freedom must be removed (six degrees of freedom for the target pose, plus one for the absolute scale) [Str11]. Following [Str11], this is achieved by fixing the corrections of two target corners to zero, i.e. $\mathbf{\Delta x_1} = (0, 0, 0)$, $\mathbf{\Delta x_2} = (0, 0, 0)$, and constraining the deviation of a third corner to the x- and y-direction only, i.e. $\mathbf{\Delta x_3} = (\Delta x_3, \Delta y_3, 0)$. Here, the points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ must not be collinear, but can otherwise be chosen freely.

2.4 Multi-view geometry and 3D reconstruction

A central application of calibrated cameras is 3D reconstruction. Given a set of images taken from different perspectives, 3D reconstruction infers the 3D structure of the scene and the camera poses associated with each image (see also App. Sec. A.6). After introducing one of the basic relations of multi-view geometry, this section introduces bundle adjustment, one of the central optimization problems in 3D computer vision. In Chap. 6, these techniques will provide the basis for self-calibration from a sequence of images.

2.4.1 Multi-view geometry

Multi-view geometry provides the basis for inferring 3D structures from two or more camera views. Let $\mathbf{I}_i$ and $\mathbf{I}_j \in \mathbb{R}^{H \times W \times 3}$ be two images taken by the

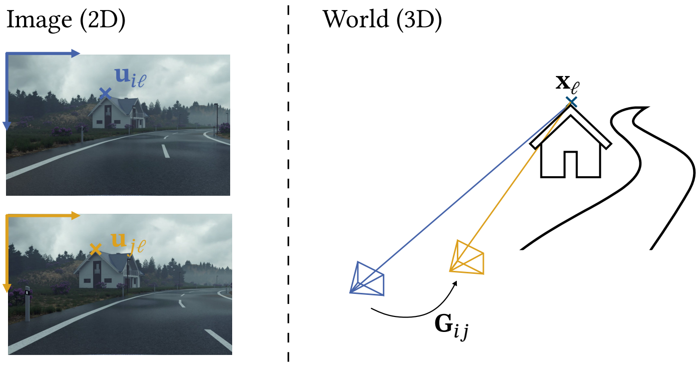


Figure 2.5: Multi-view geometry relates images captured from different viewpoints to the 3D structure of the world. The points $\mathbf{u}_{i\ell}$ and $\mathbf{u}_{j\ell}$ are corresponding points, as they reflect the same point in the 3D world.

same camera, from different views (Fig. 2.5). If the two views have an overlapping field of view, the images will contain corresponding image points, or *correspondences* $(\mathbf{u}_{i\ell}, \mathbf{u}_{j\ell})$, i.e. pairs of image points that reflect the same point $\mathbf{x}_\ell$ in the 3D world (see Fig 2.5). Assume the camera's projection is given by the function π_C and the relative pose between the two views is given by $\mathbf{G}_{\mathbf{ij}} \in SE(3)$. Then the point $\mathbf{u}_{i\ell}$ in image $\mathbf{I}_i$ can be mapped to its corresponding point $\mathbf{u}_{j\ell}$ in image $\mathbf{I}_j$ via the multi-view constraint [Hag23]:

$$\mathbf{u}_{j\ell} = \pi_C(\mathbf{G}_{ij} * \pi_C^{-1}(\mathbf{u}_{i\ell}, z_{i\ell}, \boldsymbol{\theta}), \boldsymbol{\theta}), \tag{2.20}$$

where $z_{i\ell}$ denotes the depth of the 3D point relative to view i and $\boldsymbol{\theta}$ denotes the camera's intrinsics. This relation provides the basis for inferring camera motion and 3D structure from a sequence of images.

2.4.2 Bundle adjustment

Bundle adjustment is one of the central optimization problems in 3D computer vision [Tri99]. It allows refining the estimated parameters of 3D reconstruction problems, such as Structure-from-Motion [Sch16b], visual SLAM [Cam21], or camera calibration (Sec. 2.3).

The bundle adjustment (2.18) employed during target-based calibration is a special case in which the coordinates of 3D points are known. In the following, a more general formulation of bundle adjustment is introduced, which additionally optimizes the 3D structure. This general formulation is commonly used in Structure-from-Motion and visual SLAM systems, and a variant of this formulation will be employed in Chap. 6 for intrinsic self-calibration.

Let $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{H \times W \times 3} | i = 1,...N_\mathcal{I}\}$ be a set of images and $\mathcal{X} = \{\mathbf{x}_\ell \in \mathbb{R}^3 | \ell = 1,...N_\mathcal{X}\}$ a set of 3D points. Representing the 3D structure in terms of sparse, discrete points, the bundle adjustment can be formulated as a minimization of the *reprojection error* as follows:

$$\begin{aligned}\hat{\mathbf{G}}, \hat{\mathbf{x}}, \hat{\theta} &= \underset{\mathbf{G},\mathbf{x},\theta}{\arg\min} \sum_{i=1}^{N_\mathcal{I}} \sum_{\ell=1}^{N_\mathcal{X}} ||\mathbf{u}^*_{i\ell} - \hat{\mathbf{u}}_{i\ell}||^2 \\ &= \underset{\mathbf{G},\mathbf{x},\theta}{\arg\min} \sum_{i=1}^{N_\mathcal{I}} \sum_{\ell=1}^{N_\mathcal{X}} ||\mathbf{u}^*_{i\ell} - \pi_C(\mathbf{G}_i * \mathbf{x}_\ell, \theta)||^2, \end{aligned} \tag{2.21}$$

where $\mathbf{u}^*_{i\ell} \in \mathbb{R}^2$ denotes a *measured* image point and $\hat{\mathbf{u}}_{i\ell} = \pi_C(\mathbf{G}_i * \mathbf{x}_\ell, \theta) \in \mathbb{R}^2$ denotes the corresponding *predicted* image point, given the current estimate of the 3D coordinates of the point $\mathbf{x}_\ell \in \mathbb{R}^3$, the camera pose $\mathbf{G}_i \in SE(3)$, and the intrinsics θ[1]. The vector $\mathbf{G} = (\mathbf{G}_1, ...\mathbf{G}_{N_\mathcal{I}})^\top$ contains the poses of all images in $\mathcal{I}$ and the vector $\mathbf{x} = (\mathbf{x}_1, ...\mathbf{x}_{N_\mathcal{X}})^\top$ contains the 3D coordinates of all 3D points in the reconstruction.

Bundle adjustment is a non-linear least squares problem. Therefore, optimization can be performed using the Gauss-Newton or Levenberg Marquardt algorithm introduced in Sec. 2.2.2 [Tri99].

[1] In most visual SLAM systems [Tak17, Hen15b, Hen13, Mur15, Cam21, Hen15a], the intrinsics are not optimized, but fixed to values obtained through prior calibration. However, we here include them as free parameters, as this *self-calibrating bundle adjustment* provides the basis for intrinsic camera self-calibration in Chap. 6

3 Bias detection in target-based calibration

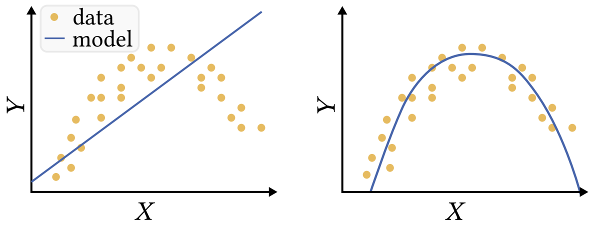


Figure 3.1: Bias in a 1D model fit.

A necessary condition for accurate calibration is that the selected model is capable of describing the true physical projection of the given lens system. If the model has insufficient degrees of freedom or is based on false assumptions on the data, the calibration will be biased.

In general, bias describes systematic deviations between the inferred model and the data (cf. Sec. 2.2, Fig. 3.1). Common causes of bias in camera calibration are insufficiently complex distortion models, and imperfections in the calibration setup [Sch20, Bec18, Str08, Cvi22, Cvi21]. Although these biases are oftentimes small, they can diminish the accuracy of downstream functions [Ozo13, Svo96, Che04, Zuc01, Che11, Abr98, Cvi22, Cvi21]. At the same time, small bias magnitudes render reliable detection difficult, and simple methods such as a visual inspection of the fit (Fig. 3.1), or computing the magnitude of residual errors, are not always sufficient, as we will show in the following.

After summarizing the state-of-the-art on detecting calibration biases, this chapter derives a method to detect biases in target-based camera calibration. To this end, this chapter adopts the approach to bias detection presented in [Hag20c, Hag22b] and incorporates corresponding experimental results. For the sake of readability, the reference to [Hag20c, Hag22b] is repeated only at key equations and results, as well as in adopted figures and tables.

3.1 State-of-the-Art

The most common approach to detecting biases is the inspection of residual errors of the calibration [Bec21, Use18, Fan22, Ope23, Mat23]. These are typically given by the deviation of *measured* image coordinates from the *predicted* image coordinates of checkerboard corners on the calibration dataset, and also called *reprojection errors* (see also (2.18)):

$$\mathbf{r}_{i,b} = \mathbf{u}^*_{ib} - \pi_C(\widehat{\mathbf{G}}_i * \mathbf{x}_b, \hat{\theta}), \tag{3.1}$$

where $\mathbf{u}^*_{ib}$ are the measured image coordinates and $\pi_C(\widehat{\mathbf{G}}_i * \mathbf{x}_b, \hat{\theta})$ are the predicted coordinates, obtained using the estimated parameters for target poses $\widehat{\mathbf{G}}_i$ and intrinsics $\hat{\theta}$. The magnitude of these errors, as quantified by the root-mean-squared-error (RMSE), is the most common metric to assess the accuracy of a calibration [Bec21, Use18, Fan22, Ope23, Mat23]:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{N_{\mathcal{J}}} \sum_{b=1}^{N_{\mathcal{X}_{\mathcal{B}}}} ||\mathbf{r}_{i,b}||^2}, \tag{3.2}$$

where n denotes the total number of observations, with each visible corner $\mathbf{u}^*_{ib}$ contributing two observations (u and v). However, as residual errors are always a superposition of measurement noise and potential biases (Fig. 3.1), it is generally not clear, which magnitude of the RMSE can be considered to be sufficiently accurate. Therefore, the RMSE is only an implicit indicator for biases.

As an alternative to the magnitude of residual errors, some works evaluate the distribution of residuals [Bec18, Sch20]. Assuming that the measurement noise follows a Gaussian distribution, significant deviations of the residual's distribution from a Gaussian distribution indicate biases. Based on this notion, [Bec18] proposes to visualize the residuals as a 2D histogram, so that deviations from a Gaussian distribution can be visually identified. Furthermore, [Sch20] proposes to compute the Kullback-Leibler divergence between the distribution of residuals and a 2D Gaussian distribution, to quantitatively assess deviations. However, as will be shown in the following, both approaches face difficulties when the magnitude of biases is small compared to the measurement noise. Furthermore, the histogram approach [Bec18] requires visual inspection which is disadvantageous when calibration should be automatized or performed by non-experts.

3.2 The Bias Ratio

In the following, we introduce a novel approach to detect calibration bias [Hag20c, Hag22b]. We argue that the main challenge in detecting biases is that they are generally superimposed by measurement noise. Therefore, the following approach introduces a way to disentangle bias from measurement noise.

In the absence of bias, assuming i.i.d. Gaussian measurement noise with variance σ^2 and maximum likelihood estimation, it can be shown that the expected value of the mean squared residual error (MSE) is asymptotically given by [Har04, p.136]

$$\mathbb{E}[\text{MSE}^\dagger] = \sigma^2 \left(1 - \frac{d^\dagger}{n^\dagger}\right), \tag{3.3}$$

where $n^\dagger$ is the number of observations, and $d^\dagger$ is the number of essential parameters. The index $\dagger$ indicates that the variables are associated with an unbiased calibration. In the presence of biases, i.e. when using insufficiently complex models, the MSE will be larger which can be expressed in terms of a superposition [Hag20c]:

$$\mathbb{E}[\text{MSE}] = \sigma^2 \left(1 - \frac{d}{n}\right) + \epsilon_{\text{bias}}^2, \tag{3.4}$$

where ϵ_{bias}^2 denotes the additional bias contribution. As neither the measurement variance σ^2, nor the bias ϵ_{bias}^2 is known a priori, the MSE provides only indirect, superimposed information on the magnitude of potential biases.

Following [Hag20c, Hag22b], we derive a way to disentangle the two contributions, to thereby determine the amount of bias in target-based calibration. The key step towards disentangling the two contributions is finding a way to separately estimate the magnitude of the measurement noise σ^2. If σ^2 is known, Eq. (3.4) can be directly solved for the bias.

To estimate σ^2, we exploit that following Eq. (3.4), for $\epsilon_{\text{bias}} = 0$, the measurement noise can be estimated as

$$\widehat{\sigma}^2 = \frac{\text{MSE}^{\dagger}}{1 - \frac{d^{\dagger}}{n^{\dagger}}}. \tag{3.5}$$

Thus, if it is possible to find a model for which $\epsilon_{\text{bias}} \approx 0$ can be assumed, the measurement noise $\widehat{\sigma}^2$ can be estimated. Then, given $\widehat{\sigma}^2$, the amount of bias in the original model can be estimated as

$$\hat{\epsilon}_{\text{bias}}^2 = \max\left\{\text{MSE} - \widehat{\sigma}^2 \left(1 - \frac{d}{n}\right), 0\right\}, \tag{3.6}$$

where $\max\{\cdot,\cdot\}$ ensures that $\hat{\epsilon}_{\text{bias}}^2 \geq 0$, which is necessary because negative values can occur due to the statistical nature of the MSE and $\widehat{\sigma}^2$.

Finally, to quantify the amount of bias in terms of a simple metric between zero and one, we compute the *bias ratio* [Hag20c, Hag22b], given by the fraction of bias in the MSE:

$$\text{BR} := \frac{\hat{\epsilon}_{\text{bias}}^2}{\text{MSE}}. \tag{3.7}$$

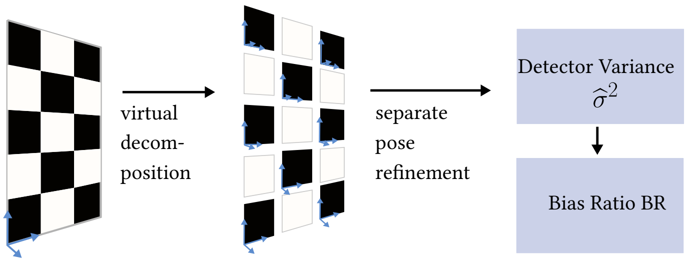


Figure 3.2: Method to estimate the bias ratio BR following [Hag20c, Hag22b]. To estimate the detector variance σ^2, the calibration target is virtually decomposed into independent local fractions, whose poses are separately re-optimized in every image. Then, the BR can be computed using (3.6)-(3.7).

If the bias ratio is close to zero, the residual errors are dominated by the measurement noise and the bias can be considered negligible. Values close to one, however, indicate relevant biases.

Having introduced the general idea of the bias ratio, we now apply it to the problem of target-based camera calibration. Specifically, a model must be found, for which the bias contribution ϵ_{bias} can be considered negligible. As potential biases can arise not only in the projection model but also in the calibration setup, it is generally not sufficient to just choose the most flexible distortion model. Instead, as shown in [Hag20c, Hag22b], we artificially construct a model that imposes as few assumptions as possible.

Specifically, we virtually decompose the calibration target (Fig. 3.2), giving fractions $\mathcal{V} = \{\text{target}_v\}_{v=1}^{N_\mathcal{V}}$, each consisting of the four corners of a checkerboard tile (Fig. 3.2). We then perform a *virtual calibration*, in which the pose of each target fraction is optimized separately, while the intrinsics are fixed to the values obtained in the original calibration. In the virtual calibration, not only the assumptions on target geometry are dropped, but also assumptions on the projection model become significantly less relevant, as each target fraction only covers a small image area. The remaining amount of bias is therefore approximated to be negligible.

Each target fraction requires estimating $d_v = 6$ parameters (rotation and translation) and each fraction consists of $n_v = 8$ measurements (four corners). Thus, the mean squared residual error MSE_v after optimizing the poses of all target fractions gives the following estimate for the measurement noise [Hag20c, Hag22b]:

$$\hat{\sigma}^2 = \frac{\mathrm{MSE}_v}{1 - \frac{d_v}{n_v}} = 4\ \mathrm{MSE}_v. \tag{3.8}$$

To reduce the impact of potential outliers, MSE and MSE_v are computed based on the median absolute deviation (MAD) as a robust estimator[1]. Finally, the bias contribution and the bias ratio can be computed as shown in Eqs. (3.6)-(3.7).

Note, that the computation of the bias ratio does not require any additional data. It uses the original calibration dataset and only drops the constraint that the checkerboard tiles form a fixed rigid body. Thereby, it can be computed after any target-based calibration without significant overhead.

3.3 Evaluation of the Bias Ratio

In the following, the validity of the bias ratio is assessed experimentally. To this end, multiple calibration datasets are collected (Fig. 3.3) and calibration is performed as described in Sec. 2.3. The dataset is also used in [Hag22b] and we adopt associated results.

3.3.1 Datasets

Simulations We simulated calibration datasets where the camera's ground-truth projection is known. To this end, we randomly sampled poses of a

[1] If a data distribution is Gaussian with sporadic outliers, the median absolute deviation (MAD) multiplied by a factor of 1.4826 gives a robust estimate of the standard deviation [Rou93, Hag22b, Hag20c].

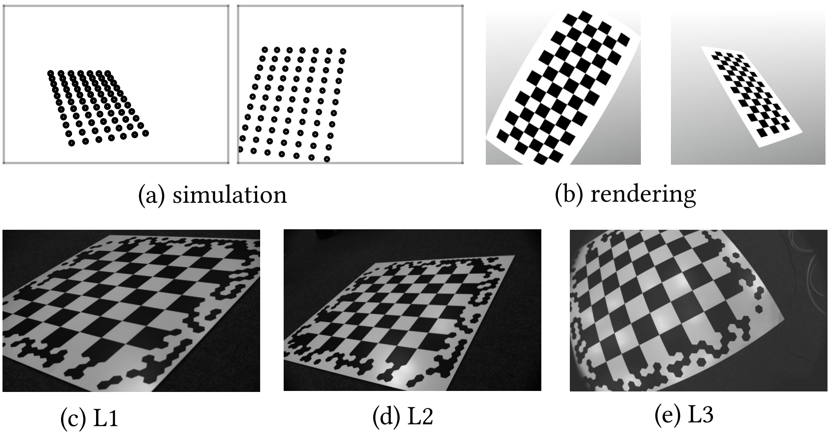


Figure 3.3: Calibration datasets used for evaluation. For the real images (L1, L2, L3), we used a single camera (Manta G-235 [Vis21]), but different lenses (XENOPLAN 1.4/17, CINEGON 1.4/12, CINEGON 1.8/4.8). The image dimensions are 4000×4000 for the simulation, 720×720 for the rendering, and 1936×1216 for all three real lenses. The same datasets are also used in [Hag22b].

checkerboard target (random rotations $\varphi_x, \varphi_y, \varphi_z \in [-\frac{\pi}{4}, \frac{\pi}{4}]$, translations $t_z \in [0.5\text{ m}, 2.5\text{ m}]$, $t_x, t_y \in [-0.5\text{ m}, 0.5\text{ m}]$) and simulated the resulting 3D coordinates of checkerboard corners in the camera coordinate system. These 3D points are projected to the image using a pre-defined camera model (Tab. D.1), giving the associated image coordinates (Fig. 3.3). To simulate measurement noise, we added random errors $\epsilon \sim \mathcal{N}(0, \sigma^2)$, $\sigma = 0.05$ px, to all image coordinates.

Rendering To account for the fact that real measurement noise might not be perfectly i.i.d. Gaussian distributed, we generated a second synthetic dataset by rendering calibration images. Thereby, the camera's projection and the target geometry are exactly known, but measurements are obtained using a real corner detection algorithm [Str14]. To this end, we generated the 3D structure of a checkerboard target and rendered calibration images through ray casting (Fig. 3.3).

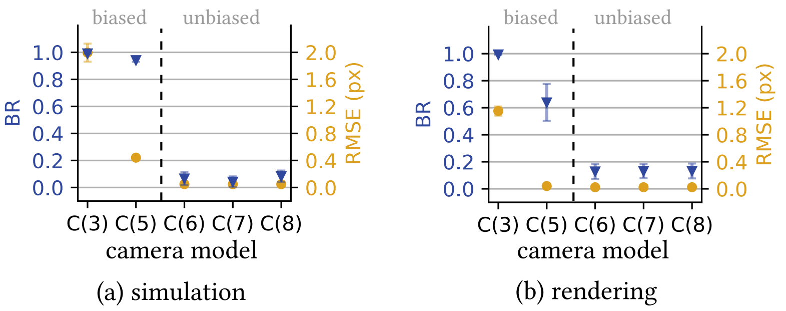


Figure 3.4: Experimental evaluation of the bias ratio. Each plot shows the bias ratio (BR) and the RMSE for different calibrations C(3)-C(8). The calibrations use models of increasing complexity, including the pinhole model (2.1) with a single focal length C(3), the radial distortion model (2.2) with one up to three distortion parameters C(5)-C(7) and the fisheye model (2.3) C(8). The BR indicates the bias in calibrations C(5)-C(8). Data points show mean and standard deviation across 10 calibrations with 50 images each. Figure adapted from [Hag22b].

Real images Finally, we collected real calibration datasets using three different lenses (Fig. 3.3). We used a coded planar checkerboard target which allows associating corners across images [Str14]. We collected 500 images for each lens and moved the camera relative to the target to obtain different perspectives.

3.3.2 Validation of the Bias Ratio

Evaluation on simulated data

The bias ratio should be close to zero for calibrations that are not biased and close to one in the presence of bias. To validate this relation experimentally, we performed both, biased and unbiased calibrations on the simulated and rendered datasets [Hag22b]. Specifically, we performed calibrations with different camera models, including insufficiently flexible ones that impose false assumptions on the data. We then computed the bias ratio (BR) and, as baseline, the RMSE (Fig. 3.4).

For insufficiently complex camera models, the BR indicates the bias with $BR \geq 0.5$ in both datasets (Fig. 3.4). In the unbiased case, the BR is significantly lower, indicating the absence of bias. In particular, this indication is more clear than using the RMSE: For model C(5), the RMSE is already comparatively low, even though the calibration is biased (Fig. 3.4). This suggests that the BR indicates small biases that are difficult to detect based on the RMSE.

However, Fig. 3.4 also shows that the BR does not consistently reach $BR = 0$ in the absence of bias. For rendered images, even unbiased calibrations yield $BR \approx 0.2$, indicating that the chosen corner detector does not produce perfectly Gaussian-distributed measurement noise. This suggests that for real-world calibration, the threshold for classifying a calibration as unbiased must be placed at $BR \geq 0$ (here $BR \approx 0.2$) to account for deviations from Gaussian measurement noise.

Evaluation on real lenses

Following [Hag22b], we further evaluate the bias ratio on real lenses. Here, the true projection is not known a priori and therefore the amount of bias of different calibrations is also unknown. Again, we performed calibration using camera models of different complexity (Fig. 3.5). Surprisingly, the BR remains comparatively high across all camera models, indicating some type of bias (Fig. 3.5, left).

Additional analyses reveal that the bias originates from imperfections in the target geometry. After precisely measuring the board geometry using a separate camera setup (using the approach described in Sec. 2.3.4), and correcting the assumed 3D coordinates accordingly, the BR of all lenses reduces to $BR \approx 0.2$ (Fig. 3.5, right). The measured non-planarity was on the order of 10^{-4} m which shows that even small inaccuracies in the calibration setup can be detected [Hag22b]. In particular, using only the RMSE, this bias could have easily been overlooked, as an RMSE of 0.1 pixels could also be attributed to the detector noise.

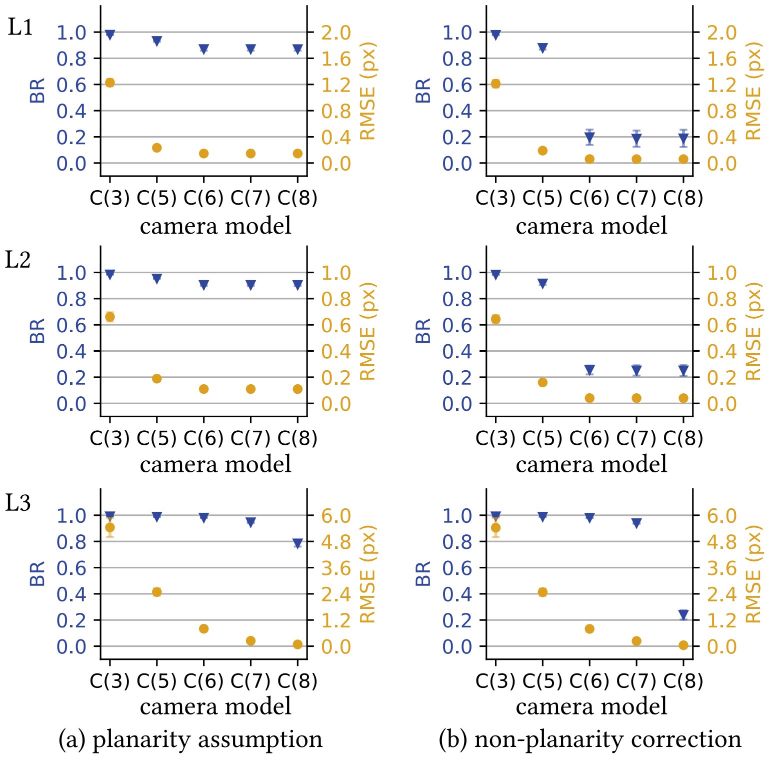


Figure 3.5: Experimental evaluation of the bias ratio (BR) for real lenses L1, L2, L3. On the x-axis, the respective projection model is annotated, including a pinhole model (2.1) with a single focal length C(3), the radial distortion model (2.2) with one up to three distortion parameters C(5)-C(7) and the fisheye model (2.3) C(8). (a) While the RMSE drops, the BR remains high across all models. (b) Taking into account the non-planarity of the target ($\Delta z \sim 10^{-4}$ m) reduces the BR significantly. Error bars are standard deviations across 10 calibrations using random samples of 50 images each. Figure adapted from [Hag22b].

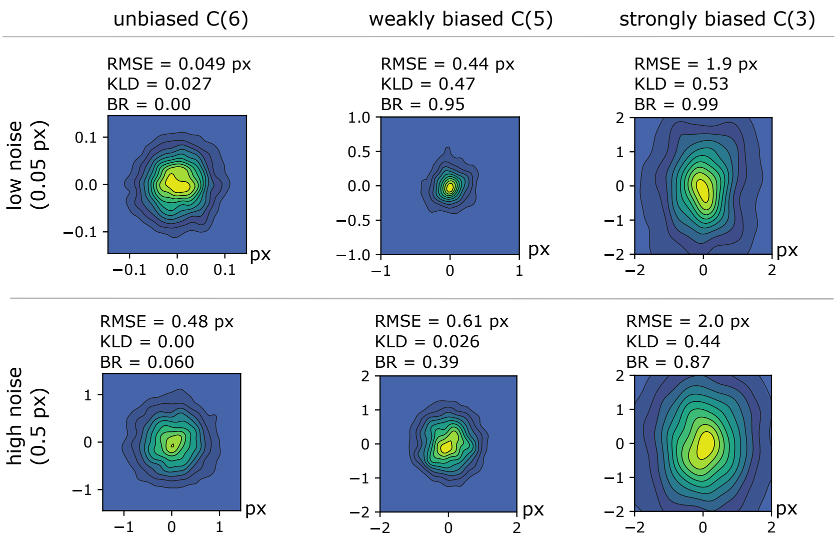


Figure 3.6: Comparison of different bias metrics on simulated data with different magnitudes of observational noise and bias. Each cell shows the 2D histogram of residuals, the RMSE, the median Kullback-Leibler divergence (KLD), and the bias ratio (BR). The same comparison for the three real lenses is shown in Fig. D.1. Figure adapted from [Hag22b].

3.3.3 Comparison of different bias metrics

Finally, we compare the different bias metrics proposed in the literature (cf. Sec. 3.1) on both, simulated and real data [Hag22b]. Specifically, we evaluate the 2D histogram of residuals [Bec18], the Kullback-Leibler divergence (KLD) between the distribution of residuals and a 2D Gaussian[1] [Sch20], the RMSE, and the bias ratio.

Across datasets, the RMSE increases with an increasing bias, yet, its absolute value depends directly on the magnitude of the measurement noise. Fig. 3.6

[1] As the KLD was initially proposed for calibration of high-dimensional camera models with significantly more observations, we had to adapt it to be applicable to our checkerboard calibration. Instead of computing the KLD within the cells of a 50x50 grid, we used a 4x4 grid.

shows that as a consequence, it is not possible to distinguish a calibration with high measurement noise from a biased calibration.

The 2D histogram of residuals [Bec18] adds more information. For the strongly biased calibrations (Fig. 3.6, right column), it clearly deviates from a circular pattern that would be expected for a 2D Gaussian. For the weakly biased calibrations, however, deviations are not as clearly visible, as the distribution is dominated by measurement noise (Fig. 3.6, middle column).

The KLD [Sch20] assesses the deviation from a 2D Gaussian quantitatively. Our results suggest that within the same dataset, the KLD increases with an increasing bias which is the intended behavior (Fig. 3.6). However, the quantitative values depend the magnitude of measurement noise, so that small biases can be overshadowed by noise (Fig. 3.6, middle column). This can be explained by the fact that for small biases, the distribution of residuals is dominated by the measurement noise.

The bias ratio also depends on the noise magnitude, yet, this is intended, as it quantifies the fraction of bias in the calibration residuals[1]. Our results suggest that the bias ratio increases with an increasing bias in both, simulations (Fig. 3.6) and real data (Fig. D.1), and that it is the only metric that correctly ranks the different calibrations in terms of their amount of bias (Fig. 3.6).

3.4 Discussion

This chapter introduces a novel approach to detecting calibration biases, called the *bias ratio* [Hag20c, Hag22b]. Experiments show that it indicates false model assumptions and, in particular, small inaccuracies in the calibration setup. Unlike existing approaches, the bias ratio does not require visual inspection and interpretation of residual error distributions or comparison with previous calibrations.

[1] To obtain the *absolute* bias, (3.6) can be used.

One limitation is that the derivation of the bias ratio relies on the ideal assumption of an i.i.d. Gaussian noise distribution. In practice, this assumption does not always hold. The results on synthetic and real lenses suggest that, as a consequence, the BR does not reach a value of zero even in presumably unbiased calibrations. Yet, some sort of assumption on the noise distribution is inevitable to distinguish noise from bias, and in accordance with previous works [Bec18, Sch20], we argue that a Gaussian distribution is a reasonable choice. The results suggest that despite the idealistic i.i.d. Gaussian assumption, the BR allows to distinguish between biased and unbiased calibrations, so that the practical applicability of the BR is not compromised by the assumption.

Overall, this chapter shows that calibration biases are oftentimes small and can easily remain undetected. However, by separately estimating the measurement noise, it is possible to infer the bias. The bias ratio can serve to select an adequate distortion model and to ensure that a calibration setup does not contain hidden imperfections, such as non-planar targets.

4 Uncertainty estimation in camera calibration

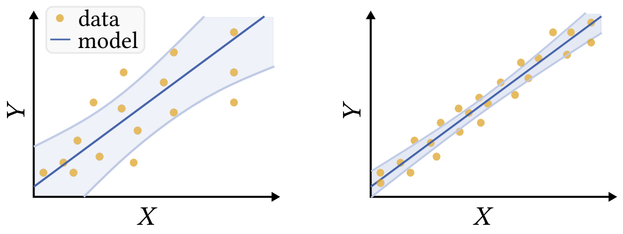


Figure 4.1: Uncertainty in a 1D model fit.

The second factor that is decisive for the accuracy of a calibration is the dataset and the resulting *precision* with which the model parameters could be estimated. The precision, or inversely the *uncertainty*, describes how much the model parameters are expected to fluctuate when performing the estimation using different data samples (Sec. 2.2). Uncertainty occurs naturally in the presence of measurement noise when the amount of data is finite.

In camera calibration, uncertainty will increase with increasing measurement noise (e.g. variance of corner detection), and with a decreasing size of the calibration dataset. Furthermore, uninformative datasets, in which the target only covers small image areas, or the target poses are indiverse, are a common cause of high uncertainty [Stu99, Pen19, Roj18, Ric13].

There are two major challenges in estimating the uncertainty of target-based calibration [Hag22b]: (i) estimating the uncertainty reliably, and (ii) expressing the uncertainty in a model-independent and easily interpretable metric. This chapter addresses both challenges.

To this end, this chapter adopts the approach to uncertainty estimation presented in [Hag20c, Hag22b] and incorporates corresponding experimental results. For the sake of readability, the reference to [Hag20c, Hag22b] is repeated only at key equations and results, as well as in adopted figures and tables.

4.1 State-of-the-Art

We distinguish between techniques to *estimate* uncertainty, and metrics to *express* the estimated uncertainty.

Uncertainty estimation Similar to machine learning methods, calibration uncertainty can be assessed indirectly in terms of a *test error*, e.g. the reprojection error on a separate test dataset [Sun06, Ric13, Sem16]. An increase in the test error compared to the training error (i.e. the reprojection error on the calibration dataset) is an indicator of uncertainty in the model. However, computing a test error requires capturing an additional test dataset which is not always feasible, and it must be ensured that the test dataset is sufficiently informative so that possible inaccuracies in the intrinsics can be captured. A further disadvantage is that the result will generally depend on the test dataset and is thereby not comparable across different calibrations.

As a direct approach to estimating uncertainty, the covariance matrix $\mathbf{\Sigma}_{\hat{\beta}\hat{\beta}}$ of estimated model parameters can be computed. As camera calibration is typically formulated as a non-linear least squares problem, the standard estimator for the covariance matrix is the backpropagation of observational noise (2.15) [Pen19, Ric13]. The matrix block $\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}$ associated with the intrinsics can then be extracted from $\mathbf{\Sigma}_{\hat{\beta}\hat{\beta}}$, giving the variances and covariances of estimated intrinsics $\hat{\theta}$. Yet, the standard estimator builds upon the assumption of i.i.d. Gaussian distributed errors. In the following, we will show that this assumption is not always valid, resulting in an underestimation of the uncertainty.

Uncertainty metrics To enhance interpretability, several works have proposed to reduce the high-dimensional covariance matrix to a scalar metric, using the $\mathrm{tr}(\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}})$ [Pen19] or the maximum index of dispersion [Roj18]. While they serve to reduce complexity, these metrics directly depend on the choice of the camera model and are thus not comparable across different calibrations. Furthermore, these metrics weight the uncertainty of each parameter equally, even though the projection function may not be equally sensitive to errors in the different parameters.

To address these issues, [Ric13] proposes the propagation of the covariance matrix to image space by Monte Carlo simulation. Specifically, the authors propose to sample intrinsics from $\mathcal{N}(\hat{\boldsymbol{\theta}}, \mathbf{\Sigma}_{\hat{\theta}\hat{\theta}})$, and to project a grid of 3D points to the image using the sampled intrinsics. The maximum standard deviation in the resulting image points is then used as an uncertainty metric, called *maxERE*. While this propagation to image space significantly enhances interpretability, a disadvantage is that it requires a Monte Carlo simulation.

As an analytically derived metric, [Str15] proposes the *observability* metric which associates a change in the intrinsics with a change in the calibration cost. The underlying idea is that if uncertainty is high, the projection model can be varied over a wide range without significantly increasing the calibration cost. The *observability* metric therefore quantifies a normalized increase in calibration cost in the least observable parameter direction $\mathbf{\Delta\theta}$, where a low observability indicates high uncertainty [Hag22b, Str15]. The derivation of the observability metric (see [Str15]) introduces important ideas, however, the metric's interpretation demands expert knowledge. Furthermore, the observability metric does not measure the overall uncertainty, but only the uncertainty associated with the least observable parameter direction, and it does not incorporate the magnitude of the measurement noise (see [Str15]).

In the following, we show that the standard covariance estimator (2.15), that all of the described metrics rely on, *underestimates* the covariance in real-world calibration, and introduce a novel, more robust covariance estimator [Hag22b]. Subsequently, we introduce a novel uncertainty metric that combines the interpretability of maxERE [Ric13] with theoretical concepts of the observability metric [Str15].

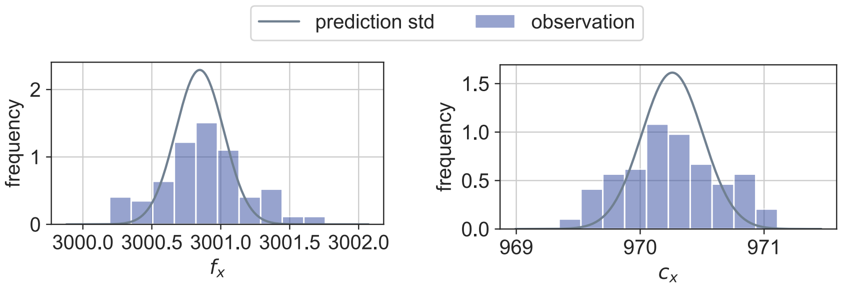


Figure 4.2: The standard covariance estimator underestimates uncertainty in real-world calibration. The histograms show the distribution of the estimated focal length f_x and principal point c_x of lens **L1** in units of pixels across multiple calibration datasets with 50 images each. The Gaussian curve shows the standard deviation predicted by the standard covariance estimator. A similar analysis is shown in [Hag22b].

4.2 Resampling-based uncertainty estimation

The standard estimator $\widehat{\boldsymbol{\Sigma}}_{\hat{\theta}\hat{\theta},\mathrm{std}}$ (2.15) for the covariance matrix in camera calibration relies on the assumption of i.i.d. Gaussian distributed measurement noise. Yet, in practice, this assumption does not always hold (see Chap. 3). Small imperfections in the calibration setup will result in a deviation from this assumption, and even in the unbiased case, the noise distribution of corner detection cannot perfectly be described by a Gaussian.

Our results on real lenses and a real corner detection algorithm suggest that the standard covariance estimator *underestimates* the variance in the intrinsics (Fig. 4.2, see also Fig. 4.8) [Hag20c, Hag22b]. Specifically, comparing the *observed* variance in estimated intrinsics across multiple calibrations with the *estimated* variance as extracted from $\widehat{\boldsymbol{\Sigma}}_{\hat{\theta}\hat{\theta},\mathrm{std}}$, the latter is smaller (Fig. 4.2). This results in overly optimistic assumptions on the quality of a calibration and poses the need for a novel, more robust uncertainty estimator.

In the following, a non-parametric approach to estimating calibration uncertainty [Hag22b] is introduced, that relies on a non-parametric statistical technique called *bootstrapping* [Dav97].

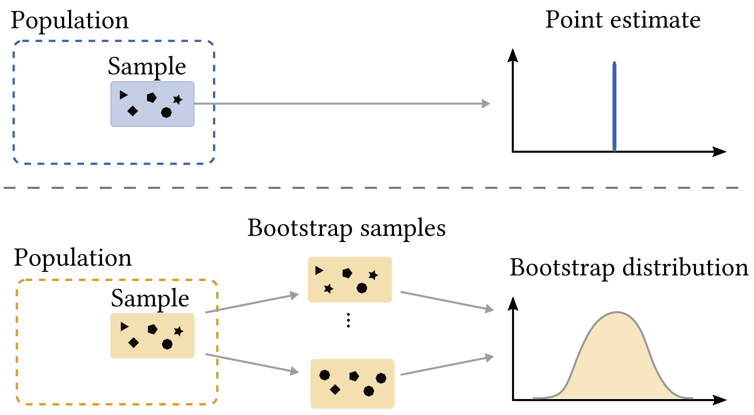


Figure 4.3: Basic principle of bootstrapping. By sampling *with replacement*, a set of bootstrap samples is obtained. The bootstrap samples have the same size as the original sample, but some elements will be missing, while others occur multiple times. The parameter of interest is estimated for each sample, giving the bootstrap distribution.

4.2.1 Bootstrapping

Bootstrapping is a non-parametric statistical technique to estimate standard deviations or confidence intervals [Dav97]. Unlike parametric approaches, such as (2.15), it does not rely on assumptions on the distribution of the data; it merely assumes that the measurement sample is representative for the population. In the following, we first explain bootstrapping for a simple one-dimensional estimation problem. Then, it is applied to the problem of camera calibration.

Let S be the available dataset consisting of n measurements, the *sample*, and β the parameter to be estimated. Assume that $\hat{\beta}$ is the point estimate for the parameter, obtained based on S. To obtain confidence intervals for $\hat{\beta}$, bootstrapping relies on *resampling* from the sample (Fig. 4.3). By sampling *with replacement* from S, bootstrapping generates a set of samples $\{S^l_{BS}\}_{l=1}^{n_{BS}}$. This set of samples is used to generate a set of point estimates $\{\hat{\beta}^l\}_{l=1}^{n_{BS}}$. The set of point estimates is called the *bootstrap distribution*. For the limit $n \rightarrow \infty$

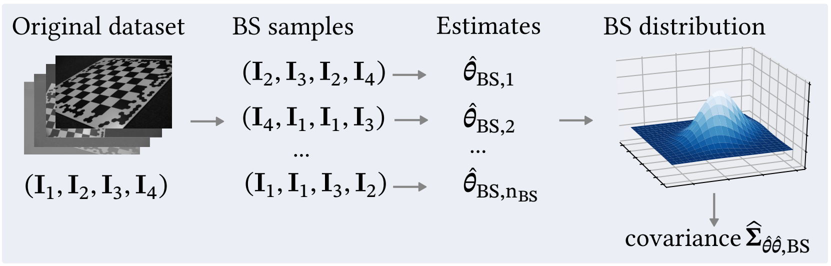

Original dataset
BS samples
Estimates
BS distribution
$(\mathbf{I}_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_4)$
$(\mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_2, \mathbf{I}_4)$
$(\mathbf{I}_4, \mathbf{I}_1, \mathbf{I}_1, \mathbf{I}_3)$
...
$(\mathbf{I}_1, \mathbf{I}_1, \mathbf{I}_3, \mathbf{I}_2)$
$\hat{\theta}_{\mathrm{BS},1}$
$\hat{\theta}_{\mathrm{BS},2}$
...
$\hat{\theta}_{\mathrm{BS},\mathrm{n_{BS}}}$
covariance $\widehat{\mathbf{\Sigma}}_{\hat{\theta}\hat{\theta},\mathrm{BS}}$

Figure 4.4: Bootstrapping uncertainty estimation for camera calibration. By sampling images with replacement from the original calibration dataset, n_{BS} bootstrap samples are generated. For each bootstrap sample, a separate calibration is performed, giving a bootstrap distribution of estimated intrinsics $\hat{\theta}$. The covariance matrix is then computed as the covariance of the bootstrap distribution. Figure adapted from [Hag22b].

and $n_{BS} \to \infty$, it can be shown that under mild assumptions, this distribution converges towards the true distribution of the estimate [Dav97]. It can therefore be used to estimate confidence intervals, standard deviations, and other statistics.

4.2.2 Uncertainty estimation using bootstrapping

As presented in [Hag22b], we propose to apply bootstrapping [Dav97] [Fox15, p.658] to the problem of camera calibration, to obtain a more robust estimator for the calibration uncertainty.

Let $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3} | i = 1,...N_{\mathcal{I}}\}$ be the set of calibration images. By sampling images with replacement from $\mathcal{I}$, we generate n_{BS} bootstrap samples $\{\mathcal{I}_{\mathrm{BS},l}\}_{l=1}^{n_{\mathrm{BS}}}$ (see Fig. 4.4). Then, we perform calibration with each of these samples, giving a set of n_{BS} estimates $\{\hat{\theta}_{\mathrm{BS},l}\}_{l=1,...n_{\mathrm{BS}}}$ of the intrinsics. This set of estimates defines the bootstrap distribution (Fig. 4.4). Finally, the variances and covariances of the intrinsics can be computed in a straightforward manner from this distribution [Hag22b]:

$$\widehat{\boldsymbol{\Sigma}}_{\hat{\theta}\hat{\theta},\mathrm{BS}} = \mathrm{Cov}(\hat{\theta}_{\mathrm{BS}}, \hat{\theta}_{\mathrm{BS}}) = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n_\theta} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n_\theta} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n_\theta 1} & \sigma_{n_\theta 2} & \cdots & \sigma_{n_\theta}^2 \end{pmatrix} . \tag{4.1}$$

with

$$\hat{\sigma}_m^2 := \frac{1}{n_{\mathrm{BS}} - 1} \sum_{l=1}^{n_{\mathrm{BS}}} \left(\theta_{lm} - \overline{\theta}_m\right)^2 \tag{4.2}$$

$$\hat{\sigma}_{nm} := \frac{1}{n_{\mathrm{BS}} - 1} \sum_{l=1}^{n_{\mathrm{BS}}} (\theta_{ln} - \overline{\theta}_n)(\theta_{lm} - \overline{\theta}_m). \tag{4.3}$$

4.2.3 Approximated bootstrapping method

The bootstrapping method avoids parametric assumptions of the classical estimator, however, it is computationally costly, as it requires performing the full calibration n_{BS} times. In the following, we will derive an efficient approximation [Hag22b] that does not require performing n_{BS} calibrations from scratch.

In Sec. 2.3, it was shown how intrinsic parameters are obtained through minimization of the reprojection error, using the Gauss-Newton or Levenberg Marquardt algorithm. Following (2.9), we can write an update step as

$$(\mathbf{J}^\top \mathbf{J})\boldsymbol{\Delta\beta} = -\mathbf{J}^\top \mathbf{r} \tag{4.4}$$

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k + \boldsymbol{\Delta\beta}, \tag{4.5}$$

where $\mathbf{r}$ are the residuals of the calibration cost (2.18), $\boldsymbol{\beta}^k = (\boldsymbol{\theta}^k, \mathbf{G}_1^k, ...\mathbf{G}_{N_\mathcal{I}}^k)^\mathsf{T}$ contains the current estimates of the intrinsics and the extrinsics, and $\mathbf{J}$ is the Jacobian of the calibration residuals w.r.t. all parameters $\boldsymbol{\beta}$[1].

In the approximated bootstrap, we make use of the fact that after convergence of the original calibration, the optimal parameters $\hat{\boldsymbol{\beta}}$, the local Jacobian $\mathbf{J}$ and the residuals $\mathbf{r}$ for the original dataset are available. We can therefore re-use them when conducting the n_BS bootstrap calibrations [Hag22b]: As for the original bootstrapping method, we construct n_BS bootstrap samples $\{\mathcal{I}_{\mathrm{BS},l}\}_{l=1,...n_\mathrm{BS}}$ by sampling with replacement from the original calibration dataset. For each bootstrap sample $\mathcal{I}_{\mathrm{BS},l}$, we now initialize the intrinsics as well as the pose parameters to the optimal values $\hat{\boldsymbol{\beta}}$ obtained from the original calibration. Furthermore, instead of re-computing the residuals and the Jacobian (which is computationally costly), we use the components of $\mathbf{J}$ and $\mathbf{r}$ from the original calibration and re-compose them, such that they contain only the entries associated with $\mathcal{I}_{\mathrm{BS},l}$. For instance, if $\mathcal{I}_{\mathrm{BS},l}$ contained the first image of the original dataset twice, the re-composed Jacobian and residuals would be given by [Hag22b]:

$$\mathbf{J}_{BS,l} = \begin{pmatrix} \mathbf{J}_1 \\ \mathbf{J}_1 \\ \mathbf{J}_3 \\ \vdots \\ \mathbf{J}_{n_\mathcal{I}} \end{pmatrix}, \quad \mathbf{r}_{BS,l} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_1 \\ \mathbf{r}_3 \\ \vdots \\ \mathbf{r}_{n_\mathcal{I}} \end{pmatrix}. \tag{4.6}$$

We then perform a single Gauss-Newton step:

$$(\mathbf{J}_{\mathrm{BS},l}^\mathsf{T}\mathbf{J}_{\mathrm{BS},l})\boldsymbol{\Delta\theta}_{\mathrm{BS},l} = -\mathbf{J}_{\mathrm{BS},l}^\mathsf{T}\mathbf{r}_{\mathrm{BS},l} \tag{4.7}$$

$$\hat{\boldsymbol{\theta}}_{\mathrm{aBS},l} = \hat{\boldsymbol{\theta}} + \boldsymbol{\Delta\theta}_{\mathrm{BS},l}, \tag{4.8}$$

giving the approximated bootstrap estimates $\hat{\boldsymbol{\theta}}_{\mathrm{BS},l}$ for each sample [Hag22b]. Finally, the covariance matrix $\widehat{\boldsymbol{\Sigma}}_{\hat{\theta}\hat{\theta},\mathrm{aBS}}$ can be estimated using (4.1)-(4.3).

[1] Note that for calibration, the Levenberg-Marquardt augmentation (2.11) is typically used, but we here use the plain Gauss-Newton formulation.

In summary, the approximated bootstrap method [Hag22b] only performs a single Gauss-Newton step for each bootstrap sample, and it re-uses the already computed residuals and local derivatives. Thereby, the computational cost is significantly reduced.

4.3 The Expected Mapping Error

In the previous sections, we have derived a non-parametric approach to estimating uncertainty in terms of the covariance matrix of model parameters. However, the covariance matrix is difficult to interpret and compare between calibrations due to the wide variety of projection models, ranging from a simple pinhole model (2.1) to local camera models with 10.000 parameters (e.g. [Sch20]) [Hag20c, Hag22b].

In the following, we therefore derive an uncertainty metric that quantifies the covariance matrix in terms of a model-independent scalar value, as shown in [Hag20c, Hag22b]. To this end, we first introduce the *mapping error*, a metric that quantifies the deviation between two projection models in image space. Then, we will show how to predict the mapping error of a calibration result $\pi_C(\mathbf{x}, \hat{\theta})$ w.r.t. the camera's true projection $\pi_C(\mathbf{x}, \bar{\theta})$, given the uncertainty in estimated model parameters $\widehat{\mathbf{\Sigma}}_{\hat{\theta}\hat{\theta}}$.

4.3.1 The mapping error

We quantify the deviation between two projection models $\pi_C(\mathbf{x}, \hat{\theta})$ and $\pi_C(\mathbf{x}, \bar{\theta})$ in image space, similar to [Bec18, Cra20, Ric13]. To this end, we define a homogeneous grid of image points $\mathcal{G} = \{\mathbf{u}_g\}_{g=1}^{n_g}$ across the given image dimension (Fig. 4.5). By applying the inverse projection $\pi_C^{-1}(\mathbf{u}_g, \bar{\theta}, z)$, points along the associated viewing rays are obtained[1]. Next, these points are backprojected to the image using $\pi_C(\mathbf{x}, \hat{\theta})$, and the mean squared deviation

[1] For central camera models, the choice of z does not influence the mapping error $\tilde{K}$. For non-central models, we propose sampling along z in a depth range of typical applications.

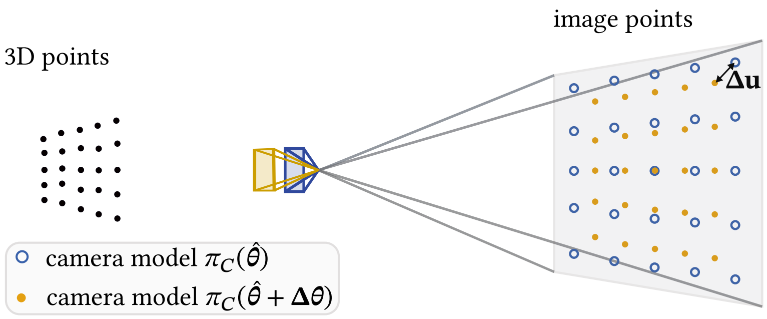


image points
3D points
$\Delta\mathbf{u}$
○ camera model $\pi_C(\hat{\theta})$
● camera model $\pi_C(\hat{\theta} + \boldsymbol{\Delta}\theta)$

Figure 4.5: The mapping error determines the average deviation between projected points of two projection models $\pi_C(\mathbf{x}, \hat{\theta})$ and $\pi_C(\mathbf{x}, \bar{\theta})$ and thereby translates a deviation in the intrinsics $\boldsymbol{\Delta}\theta = \bar{\theta} - \hat{\theta}$ into a deviation in image space. Figure adapted from [Hag22b].

between the original image points and the backprojected image points is computed (Fig. 4.5), giving the *mapping error* [Hag20c, Hag22b]:

$$\tilde{K}(\hat{\theta}, \bar{\theta}) = \frac{1}{2n_{\mathcal{G}}} \sum_{g \in \mathcal{G}} ||\mathbf{u}_g - \pi_C(\pi_C^{-1}(\mathbf{u}_g, \bar{\theta}, z), \hat{\theta})||^2, \tag{4.9}$$

where $2n_{\mathcal{G}}$ is the total number of image coordinates. If the two projection models are equal, $\tilde{K}(\hat{\theta}, \bar{\theta})$ will be zero, and it will increase with an increasing deviation between the models[1].

As a variant of this metric, we follow [Str15] and account for the fact that a camera's projection is typically preceded by a coordinate transformation from the world to the camera coordinate system. In many applications, this transformation is estimated during application and can partially compensate deviations in the intrinsics [Str15]. To account for this compensation, we define the *effective mapping error* which quantifies the mapping error after an

[1] While the two projection models $\pi_C(\mathbf{x}, \hat{\theta})$ and $\pi_C(\mathbf{x}, \bar{\theta})$ in (4.9) only differ in their intrinsics θ, the mapping error can generally also be computed based on different types of models, e.g. comparing a pinhole model with a fisheye model.

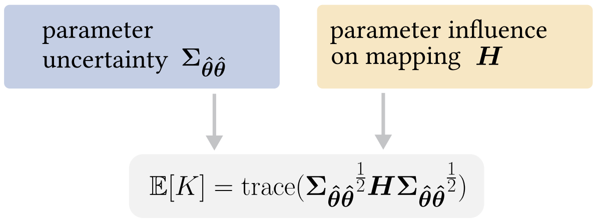

parameter uncertainty $\Sigma_{\hat{\theta}\hat{\theta}}$
parameter influence on mapping $\boldsymbol{H}$
$\mathbb{E}[K] = \mathrm{trace}(\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}} \boldsymbol{H} \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}})$

Figure 4.6: The expected mapping error (EME) weights the uncertainty in model parameters (i.e. the covariance matrix $\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}$) with the parameters' influence on the camera's mapping (i.r. the model matrix $\mathbf{H}$). By quantifying uncertainty in image space, the EME is independent of the camera model. Figure adapted from [Hag22b].

optimal compensating rotation $\mathbf{R} \in SO(3)$ [Hag20c, Hag22b]:

$$K(\hat{\theta}, \bar{\theta}) = \min_{\mathbf{R} \in SO(3)} \frac{1}{2n_{\mathcal{G}}} \sum_{g \in \mathcal{G}} ||\mathbf{u}_g - \pi_C(\mathbf{R}\, \pi_C^{-1}(\mathbf{u}_g, \bar{\theta}, z), \hat{\theta})||^2. \tag{4.10}$$

We here compensate rotation only, as this implicitly places the 3D points at infinity, and avoids the hyperparameter of choosing depths of 3D points explicitly [Hag22b]. However, one may also choose to compensate both, translation and rotation, depending on the use-case.

4.3.2 Derivation of the expected value

Given the covariance matrix $\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}$, we can predict the mapping error of a calibration result $\pi_C(\mathbf{x}, \hat{\theta})$ w.r.t. the true (unknown) projection model $\pi_C(\mathbf{x}, \bar{\theta})$. More specifically, we can predict the *expected value* $\mathbb{E}[K(\hat{\theta}, \bar{\theta})]$ which quantifies how much the calibration result is expected to deviate from the camera's true projection given the uncertainty in model parameters. This expected value is independent of the choice of the camera model and thereby serves as an uncertainty metric [Hag20c, Hag22b].

To derive $\mathbb{E}[K(\hat{\theta}, \bar{\theta})]$, we approximate the mapping error (4.10) with a Taylor expansion around $\hat{\theta} = \bar{\theta}$, inspired by [Str15]:

$$\begin{aligned} K(\hat{\theta}, \bar{\theta}) &\approx K(\bar{\theta}, \bar{\theta}) + \nabla K \boldsymbol{\Delta\theta} + \frac{1}{2} \boldsymbol{\Delta\theta}^{\mathsf{T}} \mathbf{H}_K \boldsymbol{\Delta\theta} \\ &\approx \frac{1}{2n_{\mathcal{G}}} \boldsymbol{\Delta\theta}^{\mathsf{T}} (\mathbf{J}_{\mathbf{r}_K}^{\mathsf{T}} \mathbf{J}_{\mathbf{r}_K}) \boldsymbol{\Delta\theta} \\ &\approx \boldsymbol{\Delta\theta}^{\mathsf{T}} \mathbf{H} \boldsymbol{\Delta\theta}, \end{aligned} \tag{4.11}$$

where $\boldsymbol{\Delta\theta} = \bar{\theta} - \hat{\theta}$ denotes the deviation of the estimated intrinsics from the true intrinsics, $\mathbf{r}_K$ are the residuals of the mapping error, $\mathbf{J}_{\mathbf{r}_K} = [\partial \mathbf{r}_{\mathbf{K}} / \partial \boldsymbol{\Delta\theta}]$ is the Jacobian of the mapping error and $\mathbf{H}_K$ denotes the Hessian [Hag20c, Hag22b]. The second step in (4.11) substitutes the gradient and the Hessian of $K(\hat{\theta}, \bar{\theta})$ and simplifies the resulting expression which is shown in detail in App. Sec. B.1. In the last step, we introduced the *model matrix* $\mathbf{H} := \frac{1}{2n_{\mathcal{G}}} \mathbf{J}_{\mathbf{r}_K}^{\mathsf{T}} \mathbf{J}_{\mathbf{r}_K}$ which determines the influence of $\boldsymbol{\Delta\theta}$ on the mapping error $K(\hat{\theta}, \bar{\theta})$. Given approximation (4.11), we can derive the distribution of $K(\hat{\theta}, \bar{\theta})$, as well as the expected value $\mathbb{E}[K(\hat{\theta}, \bar{\theta})]$, by propagating the covariance matrix $\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}$, as described in the following.

Assuming an unbiased model, estimated model parameters $\hat{\theta}$ obtained from a least squares optimization asymptotically follow a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_{\theta} = \bar{\theta}$ and covariance $\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}$, i.e. $\hat{\theta} \sim \mathcal{N}(\bar{\theta}, \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}})$ [Tri00, p. 8]. It directly follows that the distribution of $\boldsymbol{\Delta\theta} = \bar{\theta} - \hat{\theta}$ is given by $\boldsymbol{\Delta\theta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}})$.

Propagating this distribution to the mapping error $K(\hat{\theta}, \bar{\theta})$, we find that $K(\hat{\theta}, \bar{\theta})$ can be expressed as a linear combination of χ^2 random variables [Hag20c, Hag22b]:

$$\begin{aligned} K(\hat{\theta}, \bar{\theta}) &= \boldsymbol{\Delta\theta}^{\mathsf{T}} \mathbf{H} \boldsymbol{\Delta\theta} \\ &= \sum_{m=1}^{n_\theta} \lambda_m Q_m, \quad \text{with} \quad Q_m \sim \chi^2(1), \end{aligned} \tag{4.12}$$

where the coefficients λ_m are the eigenvalues of the matrix product $\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}\mathbf{H}\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}$ and n_θ is the number of parameters θ (for a detailed derivation, see App. Sec. B.1).

Finally, based on (4.12), the expected value of $K(\hat{\theta}, \bar{\theta})$ can be directly derived as follows [Hag20c, Hag22b]:

$$\begin{aligned}\mathbb{E}[K(\hat{\theta}, \bar{\theta})] &= \mathbb{E}[\sum_{m=1}^{n_\theta} \lambda_m Q_m] = \sum_{n=1}^{n_\theta} \lambda_m \mathbb{E}[Q_m] = \sum_{m=1}^{n_\theta} \lambda_m \\ &= \operatorname{trace}(\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}\mathbf{H}\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}),\end{aligned} \tag{4.13}$$

where we used that $\mathbb{E}[\chi^2(1)] = 1$. In the following, this expected value (4.13) will be referred to as the *expected mapping error* [Hag20c, Hag22b]:

$$\text{EME} \coloneqq \operatorname{trace}(\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}\mathbf{H}\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}), \tag{4.14}$$

and it will be used as a model-independent uncertainty metric (Fig. 4.6).

4.4 Evaluation

In the following, the resampling-based uncertainty estimation and the expected mapping error are assessed experimentally. Specifically, it is assessed whether the expected deviation from the true projection model $\mathbb{E}[K(\hat{\theta}, \bar{\theta})]$ is, on average, consistent with the true deviation $K(\hat{\theta}, \bar{\theta})$. Furthermore, the proposed methods are compared to state-of-the-art approaches from the literature. All analyses are performed on the target-based calibration datasets described in Sec. 3.3.1 and we adopt associated results from [Hag20c, Hag22b].

4.4.1 Validation of the bootstrapping method

As a first qualitative assessment of the bootstrapping method (BS) and the approximated bootstrapping method (aBS), we performed the same analysis

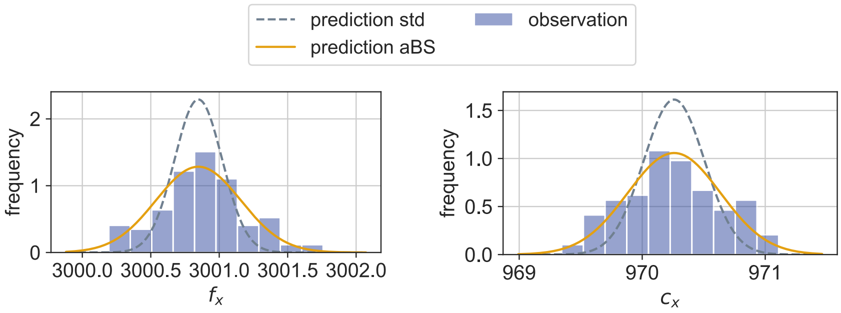


Figure 4.7: Qualitative assessment of the approximated bootstrapping method (aBS) method. The histograms show the distribution of the focal length f_x and principal point c_x of lens **L1** in units of pixels across multiple calibration datasets with 50 images each. The Gaussian curve shows the standard deviation predicted by the standard estimator and aBS method. The variance predicted by the aBS method is significantly closer to the observed distribution than the standard method.

shown in Fig. 4.2, which compares the average *predicted* distribution of intrinsic parameters with the *observed* distribution when repeatedly performing calibrations with different sets of images (Fig. 4.7). The results show that the variance predicted by the aBS method is significantly closer to the observed distribution than the standard method.

To quantitatively compare all three methods (std, BS, aBS) on multiple datasets, we used simulated and real images and compared the EME (4.13) obtained with each method to the true mapping error w.r.t. the ground truth or reference intrinsics[1] (Fig. 4.8), as presented in [Hag22b]. On ideal simulated data with i.i.d. Gaussian distributed noise, all methods yield an EME close to the true average mapping error. However, when mimicking non-ideal data by simulating bias[2], the methods differ. While the standard estimator significantly underestimates the uncertainty, the BS and aBS estimators remain close to the true mapping error. The same result can be observed with the

[1] The reference intrinsics are obtained as the average of ten calibrations based on 50 random images each and listed in Tab. D.1.

[2] To simulate bias, we simulated a projection with two non-zero radial distortion parameters, but performed calibration with only one radial distortion parameter.

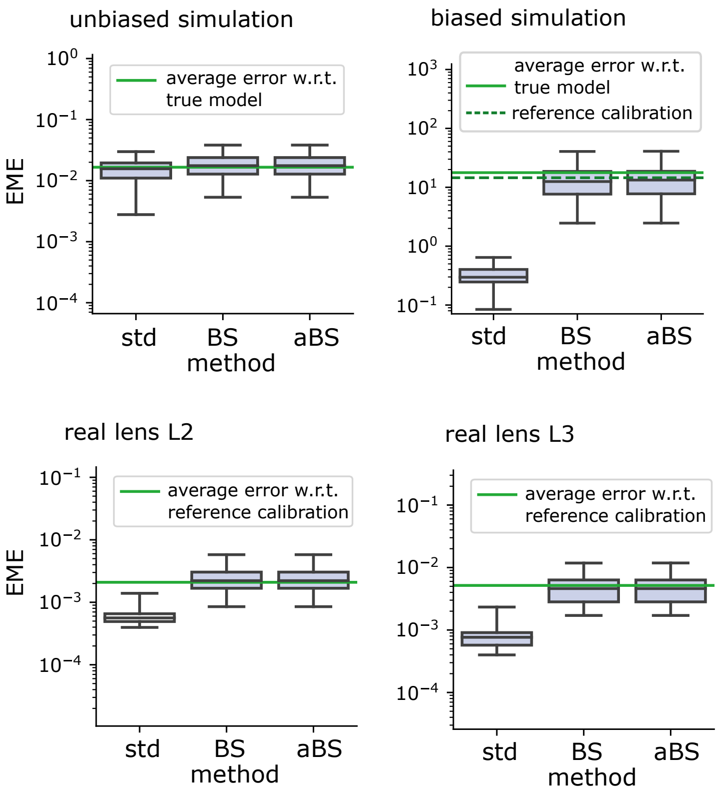


Figure 4.8: Validation of bootstrapping uncertainty estimation. Deviations from the true average mapping error (green line) show inaccurate uncertainty estimates. For ideal simulations, the standard method (std), the bootstrapping method (BS) and the approximated bootstrapping method (aBS) all yield EMEs close to the true error. In the presence of bias, and for real lenses, the standard method underestimates the uncertainty, while BS and aBS remain close to the true error. For each dataset, we used 50 random samples containing $N_{\mathcal{J}} = 25$ images each. For comparability, the BS and aBS methods use the same bootstrap samples. In the biased simulation (upper right), the true model has two distortion parameters, but calibration used only one. The dashed line therefore additionally shows the average error w.r.t the reference calibration, which also employs a single distortion parameter. Figure adapted from [Hag22b].

real lenses, showing that the real calibrations do not fulfill the assumptions underlying the standard estimator.

Overall, these results suggest that the standard covariance estimator only provides reliable estimates in ideal, simulated settings. The bootstrapping methods, on the other hand, are more robust and can be applied even for imperfect real-world calibration. Furthermore, the results suggest that the approximation of the aBS method is valid, as it produces results that are consistent with the BS method.

4.4.2 Comparison of different uncertainty metrics

To evaluate the EME and compare it with existing uncertainty metrics, we simulated calibrations of different uncertainty, as shown in [Hag22b]. Specifically, we used different dataset sizes ($N_{\mathcal{J}} \in \{10, 15, ...35\}$), as well as different magnitudes of the detector variance σ_d^2, as uncertainty is known to scale with these two quantities. Furthermore, we used informative target poses (diverse perspectives, target covering a large fraction of the image), and less informative target poses (many fronto-parallel perspectives and target appearing small in the image). Then, we estimated the covariance matrix using the aBS method[1] and computed the different uncertainty metrics, including $\mathrm{tr}(\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}})$, $maxERE$ [Ric13] and the observability [Str15] (Fig. 4.9).

Fig. 4.9 shows that all scale monotonously with the true mapping error K, which is expected, as we computed all metrics based on $\widehat{\mathbf{\Sigma}}_{\hat{\theta}\hat{\theta},\mathrm{aBS}}$. However, their magnitudes and the type of scaling differ [Hag20c, Hag22b]:

As the EME is a direct prediction of the mapping error, $\sqrt{K}$ and $\sqrt{EME}$ show approximately the same magnitude and scaling (see also App. Fig. D.2 for a direct comparison of K and EME).

[1] Note, that all metrics were originally proposed in combination with the standard covariance estimator. However, as our results suggest that the aBS estimator is more reliable, we compute all metrics based on $\widehat{\mathbf{\Sigma}}_{\hat{\theta}\hat{\theta},\mathrm{aBS}}$.

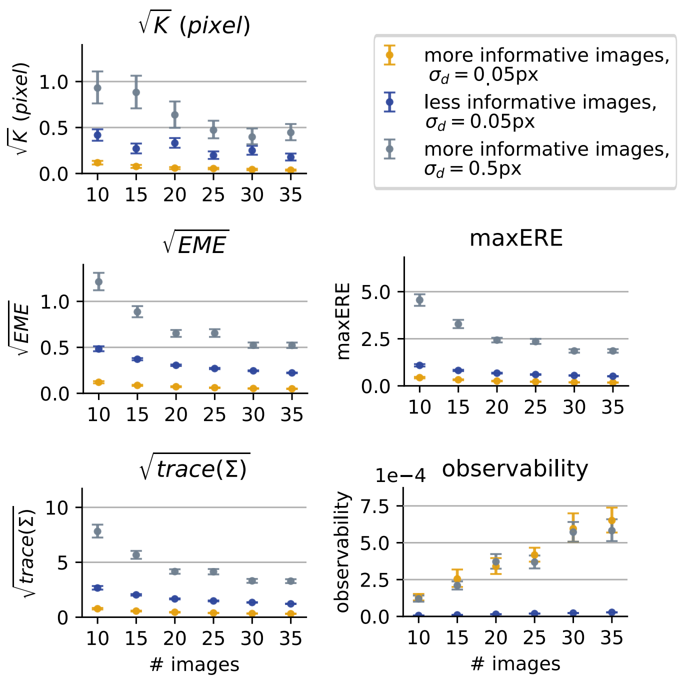


Figure 4.9: Comparison of different uncertainty metrics based on simulated calibration datasets (see legend) with different numbers of images. The upper panel shows the true mapping error $\sqrt{K}$. The metrics maxERE, $\sqrt{EME}$, and the true error $\sqrt{K}$ are all expressed in units of pixels for comparability. Plots show mean and 95% bootstrap confidence intervals across 50 random data samples. Figure adapted from [Hag22b].

maxERE also quantifies uncertainty in image space, but it uses a Monte-Carlo simulation and determines a *maximum* standard deviation in image space. Therefore, its values are generally higher than those of the mapping error $\sqrt{K}$.

The metric trace($\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}$) quantifies uncertainty in parameter space by summing the variances of all parameters. As such, it does not have a physical unit, and its magnitude will be different for different parametrizations of the model. Thereby, it cannot be compared across calibrations, limiting its applicability.

The observability metric abstracts from the camera model and is thereby comparable across models, however, its scaling is qualitatively different from the other metrics. It increases approximately linearly with the number of images, and it does not distinguish between high and low detector variance. As such, it can be interpreted as an assessment of the target-camera configurations, rather than an assessment of the calibration uncertainty as defined in Sec. 2.2 (see also [Str15]).

Overall, all metrics provide accurate information on the uncertainty, as long as they build upon a reliable estimate of the covariance matrix. The selection of the uncertainty metric is therefore a design choice. However, metrics that abstract from the chosen camera model have the benefit of being comparable across calibrations. The derived EME extends the existing metrics by quantifying uncertainty in image space without relying on a Monte Carlo simulation. Thereby, it combines the properties of maxERE [Ric13] and the observability [Str15].

4.5 Discussion

This chapter provides experimental evidence that the standard covariance estimator underestimates the calibration uncertainty in non-ideal real-world calibration, resulting in overly optimistic assumptions on the calibration accuracy [Hag22b]. As this estimator is widely used [Pen19, Roj18, Ric13], this effect is relevant to be aware of.

To obtain reliable uncertainty estimates in real, non-ideal settings, a bootstrapping method is introduced which does not rely on parametric assumptions on the data distribution [Hag22b]. Furthermore, an approximation of the method is introduced which yields comparable results with lower computational resource requirements [Hag22b]. Experiments suggest that the BS method, as well as the aBS method, provide more reliable uncertainty estimates than the standard estimator. Thereby, they may replace the standard estimator for estimating uncertainty in real-world calibration.

A limitation of bootstrapping is that it requires a sufficiently large dataset, as the theory behind bootstrapping is based on asymptotic assumptions with infinite sample sizes [Hag22b]. Our experiments suggest that this results in overly pessimistic uncertainty estimates for datasets with less than 15 images (Fig. 4.9, App. Fig. D.2). However, in practice, overly conservative uncertainty estimates are typically preferred over the overly optimistic estimates that would be obtained with the standard estimator. Therefore, the bootstrapping estimator may be preferred over the standard estimator even if the size of the dataset is small.

Finally, the expected mapping error (EME) is derived to predict the average mapping error in image space, thereby providing a model-independent uncertainty metric [Hag20c, Hag22b]. The derivation shown in Sec. 4.3 is not specific to the proposed formulation of the mapping error (4.10), and can generally be used to predict downstream errors introduced by the calibration uncertainty.

Overall, the proposed techniques enable inferring the uncertainty of an estimated projection model and thereby provide information on whether the given dataset was sufficiently informative. This can enable benchmarking and improving calibration setups, or even guide practitioners towards collecting an informative dataset, as demonstrated in [Hag20c].

5 Improvement of calibration with a deformable target model

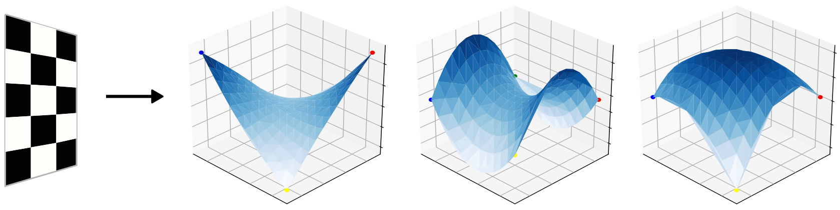


Figure 5.1: A deformable target model allows estimating and correcting deformations introduced by moving the calibration target during data acquisition. Figure adapted from [Hag22c].

The techniques presented in Chap. 3 and 4 serve to assess the accuracy of a calibration. Building upon these techniques, this chapter goes one step further and demonstrates how to improve the accuracy of target-based calibration.

Fig. 5.2 shows the bias ratio BR, obtained for two different calibrations (a) and (b) of the same camera. Although both calibrations use the same camera model and calibration algorithm, the bias ratio indicates significant bias in calibration (a), while the bias of calibration (b) can be considered negligible according to Chap. 3.

The key difference between (a) and (b) is the data collection: While calibration (a) is based on data in which the *target* was moved in front of the camera, calibration (b) is based on data in which the *camera* was moved, while the target lied statically on the ground. This reveals a relevant effect [Hag22c]: if the calibration target is manually handled, the resulting mechanical forces impact

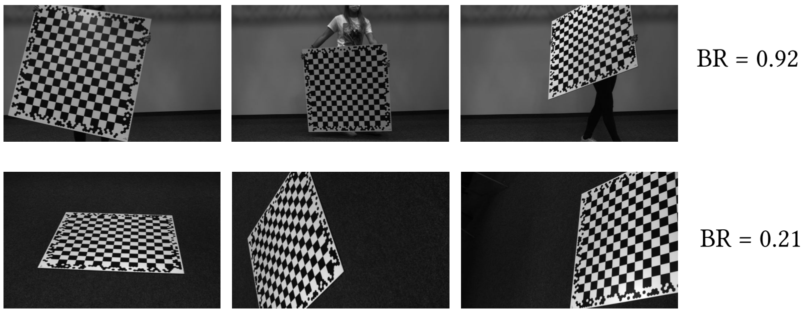


Figure 5.2: Bias ratio for different calibration datasets. Moving the calibration target in front of the camera (upper panel) results in a significantly higher BR than moving the camera around the static target (lower panel). Both datasets are acquired with the same camera and calibration is performed with the same camera model.

the target geometry and consequently the calibration. Although carrying calibration targets is standard practice [Use18, Pen19, Ric13, Bur16], this effect is typically not addressed.

To enable accurate calibration despite carrying the target, this chapter introduces a way to model such dynamic target deformations during calibration [Hag22c]. It is shown that a simple extension of the standard calibration formulation yields significantly higher calibration accuracy when moving the target.

This chapter adopts the approach to deformation modeling presented in [Hag22c] and incorporates corresponding experimental results. For the sake of readability, the reference to [Hag22c] is repeated only at key equations and results, as well as in adopted figures and tables.

5.1 State-of-the-Art

While most works on target-based calibration assume the target geometry to be accurate [Use18, Pen19, Ric13, Fan22], [Lav98] is one of the first works showing that inaccuracies in the 3D geometry of the target can result in inaccurate calibration results. To address this issue, [Lav98] proposes to estimate the 3D geometry of the target during calibration. In line with this idea, [Str08] proposes to compensate inaccuracies in the target print of planar calibration targets by estimating a 2D correction of marker positions. The idea has been further extended to optimize not just a 2D correction but to optimize the full 3D geometry of the target during calibration [Alb09], with several modifications compared to [Lav98].

Building upon these techniques, [Str11] proposes a parametrization that enables jointly optimizing intrinsics, target poses, and the target's 3D structure during bundle adjustment without ambiguities (see also Sec. 2.3.4). In experiments with imperfect checkerboard targets, a significant improvement in calibration accuracy was achieved [Str11].

However, all of the existing approaches [Lav98, Str08, Str11, Hua13, Xia17] focus on *static deviations* from the assumed target geometry. This means that they only allow for a single correction of the target geometry which is assumed to be constant across all calibration images. *Dynamic deviations*, which change in every image, cannot be modeled with these approaches.

5.2 Dynamic deformation model

Unlike static deformations (Sec. 2.3.4), dynamic deformations require different corrections $\mathbf{\Delta x}_{ib}$ of 3D target coordinates in every image $\mathbf{I}_i$. However, allowing for a free correction of 3D points in every image would result in an unconstrained optimization problem. Following [Hag22c], we therefore propose to constrain the optimization by exploiting *spatial correlations* in the displacements of 3D points.

Specifically, we propose to exploit that physical deformations of a rigid target will result in spatially correlated displacements of points on the target (Fig. 5.1). As a consequence, the displacements can be approximated by a low-dimensional function $\mathbf{\Delta x}_{ib} = \mathbf{f}(\mathbf{x}_b, \boldsymbol{\gamma}_i)$ which only depends on a small set of model parameters γ_i per image i. These parameters can be estimated jointly with camera poses and intrinsics during calibration.

Given a deformation model $\mathbf{f}(\mathbf{x}_b, \gamma_i)$, we formulate the optimization by extending (2.18) as follows [Hag22c]:

$$\hat{\theta}, \widehat{\mathbf{G}}, \hat{\gamma} = \underset{\theta, \mathbf{G}, \gamma}{\arg\min} \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{C}} ||\mathbf{u}^*_{ib} - \pi_C(\mathbf{G}_i * (\mathbf{x}_b + \mathbf{f}(\mathbf{x}_b, \gamma_i)), \theta)||^2, \tag{5.1}$$

where θ are the intrinsics, $\mathbf{G} = (\mathbf{G}_1,...\mathbf{G}_{n_\mathcal{I}})^\top$ are the target poses and $\gamma = (\gamma_1,...\gamma_{n_\mathcal{I}})^\top$ are the deformation parameters which characterize the deformation in each image $\mathbf{I}_i$.

In general, the choice of the deformation model $\mathbf{\Delta x}_{ib} = \mathbf{f}(\mathbf{x}_b, \gamma_i)$ depends on the shape, size, and material of the target, and a possible approach is to consider the mechanical properties of the target and to physically model possible deformations [Hag22c]. However, to investigate the general idea of dynamic deformation modeling, we here use a more simplistic model. For planar checkerboard targets, we follow [Hag22c] and only model out-of-plane deformations, which can be introduced by gravity or manual handling, using a simple paraboloid (see also Fig. 5.1):

$$\mathbf{f}(\mathbf{x}_b, \gamma_i) = \begin{bmatrix} 0 \\ 0 \\ a_i x_b^2 + b_i y_b^2 + c_i x_b y_b \end{bmatrix}, \tag{5.2}$$

with $\gamma_i = (a_i, b_i, c_i)^\top, a_i, b_i, c_i \in \mathbb{R}$. We here assume the origin of the target coordinate system to be in the center of the checkerboard, so that the vertex of the paraboloid is at the center of the target. Importantly, formulation (5.2) does not result in ambiguities with the target poses, so that optimization (5.1) remains unambiguous[1].

[1] For instance, adding a constant term in (5.2) would result in ambiguity with the translation z-direction, so that the translation could not be uniquely estimated.

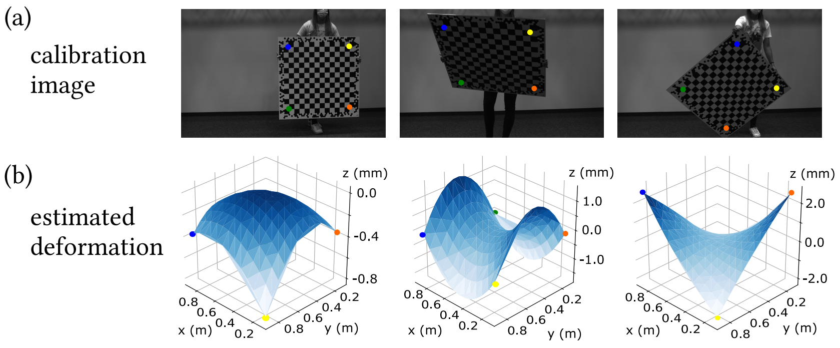


Figure 5.3: Estimated deformations for exemplary calibration images. The colored dots indicate the corresponding points and the z-axis points towards the back of the board. The estimated deformation is consistent with the intuition of how gravity is acting on the target. Figure adapted from [Hag22c].

Combining static and dynamic deformation

The proposed deformation model (5.2) only allows for an out-of-plane deformation. In practice, however, such dynamic out-of-plane deformations may be superimposed by static deformations, e.g. due to imperfect target prints. As an extension of (5.1), we therefore propose to allow for both, a static in-plane and dynamic out-of-plane correction [Hag22c].

To include a static in-plane correction $\mathbf{\Delta x}_b^{\text{static}} = (\Delta x_b^{\text{static}}, \Delta y_b^{\text{static}}, 0)^\mathsf{T}$, we follow [Str11], but adapt their ideas from 3D to 2D. For an unambiguous estimation in 2D, four degrees of freedom (DOF) must be fixed[1]. This is achieved by setting $(\Delta x_1^{\text{static}}, \Delta y_1^{\text{static}}) = \mathbf{0}$ which fixes the position of the target, and $(\Delta x_2^{\text{static}}, \Delta y_2^{\text{static}}) = \mathbf{0}$ which fixes the scale of the target and its orientation within the $z = 0$ plane.

[1] Two DOF to fix the target's position in x- and y-direction, one DOF to fix the rotation around the z-axis, and one DOF to fix the target's scale.

Finally, the optimization problem with both in-plane and dynamic out-of-plane correction is given by [Hag22c]:

$$\hat{\theta}, \widehat{\mathbf{G}}, \hat{\gamma}, \widehat{\mathbf{\Delta x}}^{\text{static}} = \underset{\theta, \mathbf{G}, \gamma, \mathbf{\Delta x}^{\text{static}}}{\arg\min} \sum_{i \in \mathcal{I}} \sum_{b \in \mathcal{C}} ||\mathbf{u}^*_{ib} - \pi_C(\mathbf{G}_i * (\mathbf{x}_b + \mathbf{f}(\mathbf{x}_b, \gamma_i) + \mathbf{\Delta x}_b^{\text{static}}), \theta)||^2. \tag{5.3}$$

5.3 Evaluation

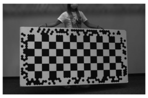
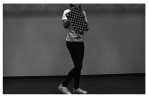
In the following, we quantitatively assess the effect of dynamic target deformations, by comparing four different calibration formulations [Hag22c]:

- **standard**: Commonly used calibration formulation that assumes a perfect, planar target (2.18),
- **static**: Extended formulation which allows for a static correction of the target geometry (2.19),
- **dynamic**: Proposed formulation that uses a deformable target model (5.1),
- **full**: Most flexible formulation, allowing for both, a static in-plane deformation and a dynamic out-of-plane deformation (5.3).

5.3.1 Datasets

We use the same dataset as also used in [Hag22c]: We collected calibration datasets using four different targets of different size and material (Fig. 5.4). During data acquisition, the targets were manually moved, as common practice in computer vision [Use18, Pen19, Ric13, Bur16]. Using an Allied Vision camera MG-235B with XENOPLAN 1.4/17 lens, we collected a total of 500 calibration images with each target. To conduct statistical analyses, we randomly sampled 50 subsets, containing 25 images each, for each analysis. In addition to the four test datasets, we collected a reference dataset consisting of 650 images with target T2 lying statically on the ground, in which dynamic deformations can be ruled out.

Table 5.1: Exemplary images and maximum estimated out-of-plane deformation $\Delta z_{\max}$ of targets **T1**-**T4**. Values show mean and standard deviation across 50 image subsets for each target. Table adapted from [Hag22c].

Target	**T1**	**T2**	**T3**	**T4**
				
Size	$100 \times 200 \times 4$ cm	$100 \times 100 \times 0.6$ cm	$60 \times 42 \times 0.7$ cm	$37.5 \times 25 \times 0.25$ cm
Material	styrofoam composite	aluminum polyethylene composite	paper glued to cardboard	aluminum polyethylene composite
$\Delta z_{\max}$	(2.6 ± 1.2) mm	(1.7 ± 2.4) mm	(3.0 ± 2.1) mm	(0.19 ± 0.57) mm

5.3.2 Accuracy of estimated intrinsics

To compare the different calibration methods *standard*, *static*, *dynamic* and *full*, we first evaluate the RMSE (3.2) obtained on all four datasets (Fig. 5.4 (a)), as shown in [Hag22c]. Across all datasets, the RMSE is reduced by modeling target deformations. However, this is not surprising, as deformation modeling introduces additional degrees of freedom into the optimization. Interestingly, however, the *dynamic* model yields lower RMSE than the *static* model for targets T1 and T2, although the *dynamic* model introduces fewer degrees of freedom. This suggests that the dynamic model better captures the underlying deformations.

As the RMSE on the calibration data is only a training error (cf. Sec. 2.2), we additionally evaluated a *test error*, as shown in [Hag22c]. To this end, we used the reference dataset with target T2 lying statically on the ground[1] and computed the test error as follows [Ric13]: Using the intrinsics obtained in

[1] As the previously conducted reference calibration (on all 650 reference images, using the *static* method) had revealed a static target deformation in the reference dataset, we included the estimated correction terms $\widehat{\boldsymbol{\Delta}\mathbf{x}}^{\text{static}}$ when computing the test error.

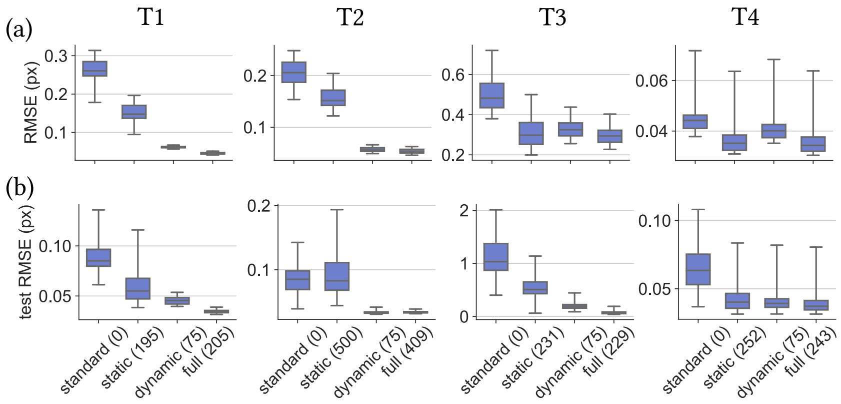


Figure 5.4: Comparison of the different calibration methods *standard*, *static*, *dynamic* and *full* for different targets T1, T2, T3, T4. The numbers in brackets are the number of deformation parameters of each method. (a) Reprojection error (RMSE) on the respective calibration datasets. (b) RMSE on a separate test dataset. Box plots include the results of 50 random subsets for each target, each containing 25 calibration images. Figure adapted from [Hag22c].

the respective calibration under test, we optimized only the target poses of the reference dataset. The final RMSE on the reference dataset then defines the test error. Across all targets, the dynamic deformation model leads to a significantly reduced test error, with either the *dynamic* or the *full* method resulting in the lowest test RMSE (Fig. 5.4).

Finally, we compare the bias ratio and the expected mapping error (based on the approximated bootstrap) of all calibrations, as proposed in Chap. 3-4. The bias ratio clearly shows the bias of targets T1-T3 when using the standard model (Fig. 5.5 (a)). Furthermore, the BR confirms that bias can be reduced by modeling deformations. For targets T1 and T4, negligible bias can be achieved when using the full deformation model. For target T2, however, a small bias remains which indicates that the parabolic deformation model does not fully capture the underlying deformation of this target. Similarly, the self-made target T3 results in a biased calibration despite deformation modeling which is expected for a low-quality self-made target. The EME additionally shows

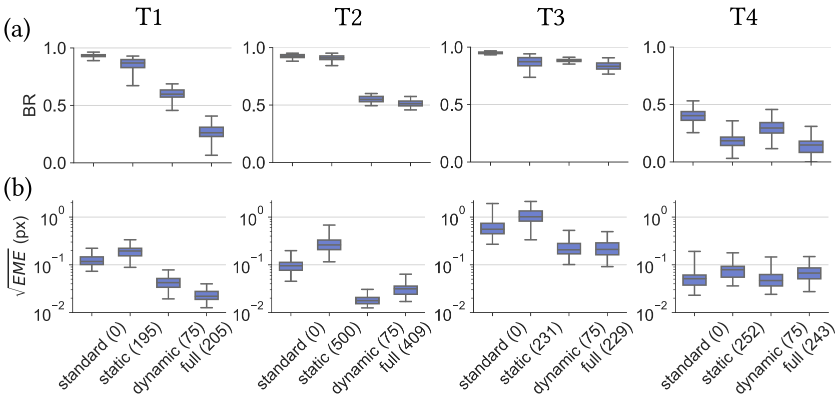


Figure 5.5: Bias and uncertainty of the different calibration methods (*standard*, *static*, *dynamic* and *full*) for different targets T1, T2, T3, T4. The numbers in brackets are the number of deformation parameters of each method. (a) Bias ratio BR. (b) Square root of the expected mapping error EME. Box plots include the results of 50 random subsets for each target, each containing 25 calibration images.

that modeling static deformations increases uncertainty, caused by the additional degrees of freedom (Fig. 5.5 (b)). Furthermore, the EME shows that the calibrations with target T4, although exhibiting low bias, exhibit higher uncertainty than the other unbiased calibrations. This can be explained by the fact that the target is small and only covers a small fraction of the image.

Overall, Figs. 5.4-5.5 show that dynamic deformation modeling leads to an improvement in calibration accuracy across all four targets. The effect is larger for targets with a larger physical extent, which is consistent with the expectation that larger targets exhibit larger deformations (see also Tab. 5.1). Consistent with Chap. 3-4, Fig. 5.5 further shows that the BR and the EME provide relevant additional information by explicitly quantifying bias and uncertainty.

Fig. 5.3 shows exemplary deformations estimated with the dynamic deformation model. The estimated shapes are consistent with the expectation of how the mechanical forces act on the target.

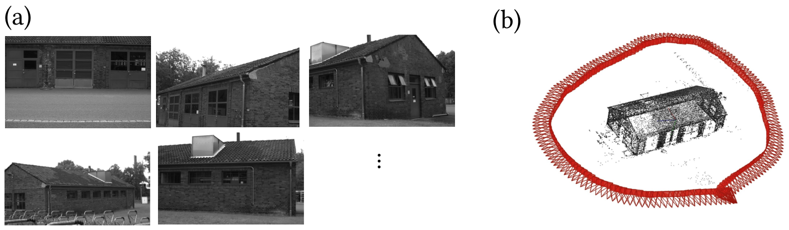


Figure 5.6: Structure-from-Motion experiment, where the camera (red pyramids) was moved around a building. Figure adapted from [Hag22c].

5.3.3 Relevance in practical applications

Finally, we assessed whether the observed differences between the calibration methods are relevant in practice. To this end, we devised a Structure-from-Motion experiment [Hag22c]. We used the same camera as in Sec. 5.3.2 and collected a dataset in which the camera moves around a building (Fig. 5.6). Using classical Structure-from-Motion (Sec. 2.4), we then reconstructed the 3D structure of the building, as well as the camera pose associated with each image. As the reconstruction relies on the previously estimated intrinsics, inaccuracies in the calibration are expected to impact the accuracy of the reconstruction.

To assess the accuracy of the reconstruction, we use a loop closure error, as proposed in [Pen19]: The first image of the sequence is added once again as the last image so that the ground-truth poses of the first and the last image are identical. During SfM, keypoints are only matched with neighboring images (sequential matching), and loop closure detection is disabled, so that potential calibration errors will accumulate along the trajectory. The loop closure error can then be computed by taking all reconstructed 3D points associated with the first image, and projecting them into the last image given their relative pose estimated during SfM. The root mean squared deviation between the corresponding points in the first and last image then yields the loop closure RMSE in image space.

Table 5.2: Loop closure RMSE in the Structure-from-Motion experiment, quantified in image space. All reconstructions were performed with AliceVision Meshroom [Ali21], using default hyperparameters. The values are mean and standard deviation across 50 calibrations. Table extracted from [Hag22c].

Target	**Calibration method**	**Loop closure RMSE (px)**
T1	standard	16.1 ± 5.7
	static	5.1 ± 3.7
	dynamic (ours)	3.5 ± 1.8
	full (ours)	$\mathbf{1.5 \pm 0.7}$
T2	standard	9.2 ± 5.0
	static	11.8 ± 7.6
	dynamic (ours)	$\mathbf{1.4 \pm 0.6}$
	full (ours)	1.8 ± 0.6
T3	standard	248.7 ± 109.3
	static	49.8 ± 31.5
	dynamic (ours)	14.2 ± 9.9
	full (ours)	$\mathbf{8.0 \pm 5.6}$
T4	standard	9.7 ± 3.2
	static	5.1 ± 2.8
	dynamic (ours)	$\mathbf{3.6 \pm 2.4}$
	full (ours)	$\mathbf{3.6 \pm 2.4}$

We performed SfM reconstructions with all calibrations obtained in Sec. 5.3.2 (obtained with different targets and calibration methods), and compared the loop closure error (Tab. 5.2). Across all targets, the calibration results obtained with the dynamic deformation model result in significantly lower loop closure errors (Tab. 5.2). This shows that dynamic target deformations are not a negligible effect, but relevant in practice.

5.4 Discussion

This chapter shows that moving a calibration target during data acquisition can lead to deformations that introduce significant inaccuracies into the inferred projection model. Although it is common practice to carry calibration targets [Use18, Pen19, Ric13, Bur16], to our knowledge this effect has not been addressed in works prior to [Hag22c].

To achieve accurate calibration despite moving the target, a deformable target model is introduced [Hag22c]. By exploiting spatial correlations between the displacements of target points, the deformation model can be estimated during calibration, adding only a small set of parameters during bundle adjustment. Experiments on four different targets show that this results in a significant improvement in calibration accuracy, so that accurate calibration can be achieved despite moving the target.

The parabolic deformation model is only a coarse approximation, and clearly, more sophisticated models could be used. For instance, one could explicitly consider the mechanical properties of the target and physically model its expected shape depending on the forces that are applied [Hag22c]. While this is an interesting future direction, the experiments suggest that a simple paraboloid already leads to significantly more accurate calibration results, while being easy to understand and implement. The proposed deformation model has meanwhile been used to re-calibrate the widely used Kitti and EuRoC datasets, giving significant improvements in calibration accuracy [Cvi22].

6 Deep learning based intrinsic camera self-calibration

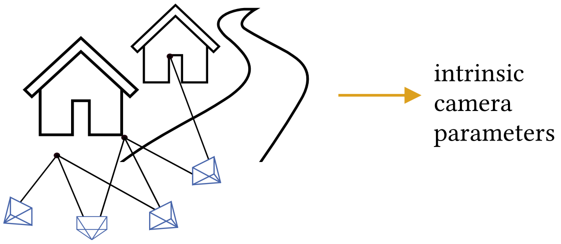


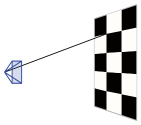
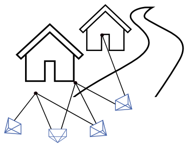
Figure 6.1: Camera self-calibration infers intrinsic camera parameters without calibration targets, based on image sequences of the environment.

The previous chapters have addressed target-based camera calibration, as it is still the widely used approach in research and industry. The main reason for using calibration targets is that they provide a controlled environment with known geometry, which enables highly accurate calibration. However, target-based calibration is limited by requiring a dedicated calibration dataset which renders, e.g., continuous re-calibration impossible (see also Tab. 6.1).

Camera self-calibration, on the other hand, infers camera parameters during application, based on images or image sequences of an unknown environment (Fig. 6.1). Thereby, it avoids the laborious calibration process and enables correcting inaccuracies during usage, which can occur, for instance, due to temperature changes or mechanical impact [Daa19, Smi10, Hay21, Cra20]. Yet, camera self-calibration is challenging because the environment in which it has to function cannot be controlled (see also Tab. 6.1). Therefore, obtaining a robust and accurate self-calibration has been a long-standing research goal [Fau92, Fan22].

After summarizing the state-of-the-art of self-calibration, this chapter introduces a novel approach to intrinsic camera self-calibration, which combines recent advances in deep learning with classical bundle adjustment. To this end, this chapter adopts the approach to intrinsic camera self-calibration presented in [Hag23] and incorporates corresponding experimental results. For the sake of readability, the reference to [Hag23] is repeated only at key equations and results, as well as in adopted figures and tables.

Table 6.1: Comparison of target-based and targetless approaches to camera calibration.

	Advantages	**Disadvantages**
Target-based 	• Controlled environment with known geometry. • Fast and repeatable on an industrial scale. • Tractable optimization problem because of prior knowledge on the environment.	• Requires specialized equipment. • Laborious process if done manually. • Cannot be repeated once the camera is deployed.
Targetless 	• Requires no equipment. • Enables re-calibration during application. • Enables estimating the intrinsics for videos without access to the camera.	• Accuracy depends on environment, motion, and lighting. • Challenge of critical motion and structureless environment. • Complex optimization problem, typically subject to outliers and a larger number of unknown parameters.

6.1 State-of-the-Art

Approaches to intrinsic camera self-calibration can generally be divided into video-based approaches, which estimate camera intrinsics by exploiting the consistency of the scene structure over time, and single-image approaches, which use only a single image taken by the camera and rely on assumptions on the structure of the scene.

6.1.1 Self-calibration from video

The first approaches to intrinsic camera self-calibration used a sequence of images with overlapping field of view and exploited the consistency of the scene structure across images [Fau92, May92, Hem03]. The mathematical basis was presented in [Fau92, May92], where the properties of the *absolute conic* and its image (the *image of the absolute conic, IAC*) are used to derive a method for estimating the camera matrix **K** of a perspective camera (cf. Sec. 2.1). Many subsequent works have built upon these geometric relations, and are summarized in [Hem03].

As an alternative to these principled approaches, and to enable the self-calibration of cameras that cannot be described by a plain pinhole model, subsequent works proposed to perform self-calibration within a Structure-from-Motion [Tri99, Sch16a] or visual SLAM [Civ09, Kei15] pipeline. This can be achieved by jointly optimizing 3D structure, poses, and intrinsics during bundle adjustment (see Sec. 2.4.2, Fig. 6.2 (a)). Although such approaches depend on a reasonable initial guess for the intrinsics, they have proven effective in allowing for different camera models, including lens distortion. However, there is little literature on the accuracy of estimated intrinsics, and most visual SLAM systems still rely on prior target-based calibration [Tak17, Hen15b, Hen13, Mur15, Cam21, Hen15a].

More recently, the classical approaches to self-calibration have been complemented by deep learning approaches [Yan17, Bog18] (see also App. Sec. A.1). While most deep learning approaches aim at single-image self-calibration (Fig. 6.2 (d)), there also exist approaches that utilize image sequences

(Fig. 6.2 (b)). These approaches typically do not aim solely at self-calibration, but perform depth estimation [Gor19, Che19] or learn a neural radiance field (NeRF) representation [Jeo21], based on videos from unknown cameras. Estimating the intrinsics is therefore a means to solving a different task.

In [Che19], a convolutional neural network (CNN) architecture is used to simultaneously regress pixel-wise depth, optical flow, relative camera pose, and intrinsics from a pair of images (Fig. 6.2 (b)). Simultaneously estimating depth, pose and intrinsics allows formulating a self-supervised loss function using a photometric error [Che19], enabling training without labeled data. This general idea is also used in [Gor19], containing several modifications in the network architecture and loss function. However, experimental evaluation has shown that the intrinsics predicted for pairs of images fluctuate significantly, as they are learned in aggregate with the pose parameters [Gor19].

Therefore, [Gor19] additionally proposes an alternative to the regression of intrinsics: Instead of making the intrinsics an output of the network, the intrinsics can be implemented as learned parameters (i.e. weights) of the model (Fig. 6.2 (c)) [Gor19]. Thereby, the intrinsics of a specific camera can be learned during training. This general idea has been extended to high-dimensional camera models [Vas20] and the unified camera model (see Sec. 2.1) [Fan22] and it was shown that it enables comparatively accurate depth estimation without prior knowledge of the intrinsics [Fan22]. However, a disadvantage of implementing intrinsics as learned parameters is that the trained model will be restricted to the camera from the training data, and switching to a different camera will require re-training the model.

Finally, besides self-calibration approaches that rely only on the consistency of the scene across views, there also exist approaches that make use of additional prior knowledge on the structure of the scene or the camera motion. These include the usage of planar constraints [Tar07], pre-built 3D maps [Lin20], or exploiting purely rotational camera motion [Har94].

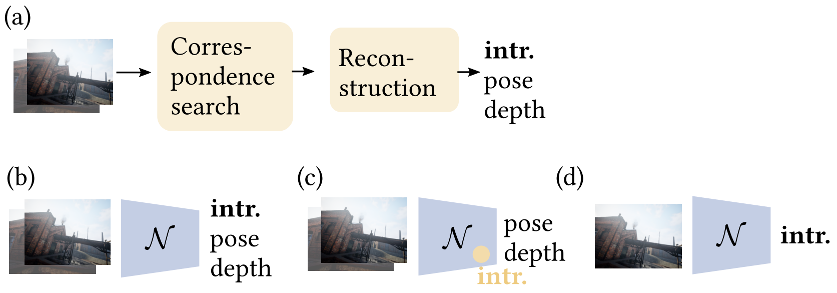


Figure 6.2: Schematic depiction of different approaches to intrinsic camera self-calibration in the literature. (a) Classical Structure-from-Motion [Sch16a] with self-calibrating bundle adjustment (see also App. Sec. A.6), (b) a deep neural network jointly inferring camera pose, pixel-wise depth and intrinsics based on a pair of images [Gor19], (c) a deep neural network containing the intrinsics as learned parameters that are optimized during training [Fan22], and (d) single-image self-calibration [Lop19]. Here $\mathcal{N}$ symbolizes a deep neural network. Figure adapted from [Hag23].

6.1.2 Single-image self-calibration

If the content of an image fulfills certain assumptions, a single image can be sufficient to estimate intrinsic camera parameters (Fig. 6.2 (d)). For instance, given vanishing points in the image, whose vanishing lines are known to be perpendicular (e.g. lines along the perpendicular walls of a building), the focal length can be estimated [Cip99]. Furthermore, assuming the straightness of lines in manmade environments allows inferring lens distortion because distortions will map straight lines to curved lines in the image [Ant17].

These basic ideas have been exploited by classical handcrafted algorithms, but also by deep learning approaches [Cip99, Bog18, Lop19, Yin18, Xue19, Lee21]. There exist approaches to image undistortion where a deep neural network infers the undistorted version of a fisheye image [Yin18, Xue19]. Furthermore, several works have proposed to regress focal length, radial distortion, and horizon line from single images, using deep neural networks [Bog18, Lop19]. As one of the more recent works, [Lee21] proposes to divide the self-calibration task into steps and to first estimate line segments in the image, which are subsequently processed by a transformer network to predict the intrinsics. Finally, [Zhu19] proposes to integrate a single-image self-calibration network into a SLAM system to deal with unknown cameras in visual SLAM.

The disadvantage of single-image approaches for self-calibration lies in the strong assumptions about the scene and perspective (e.g. that lines should be straight [Xue19] or that certain objects contain right angles [Cip99]). Furthermore, most single-image approaches can only infer a limited subset of the intrinsic parameters (e.g. estimating only the distortion, or assuming the principal point to be in the image center [Lop19]). The method proposed in this thesis therefore relies on a sequence of images which allows using multi-view constraints rather than assumptions on the image content.

6.2 Self-calibrating bundle adjustment as a network layer

The self-calibration approach introduced in the following infers a camera's intrinsics θ from monocular video by exploiting the consistency of the scene structure over time [Hag23]. The rationale behind the approach is to leverage the capabilities of deep learning, but instead of relying on a purely learned model, we explicitly implement projection functions and multi-view geometry. This way, the model does not need to learn the well-known underlying multi-view geometry, and unlike pure deep learning approaches (Sec. 6.1.1), it is not tied to a specific projection model.

As the approach builds upon recent research on combining deep learning with classical bundle adjustment, the following section first introduces the related work in this field. Then, the proposed self-calibration approach, which is also presented in [Hag23], is described conceptually, focusing on the self-calibrating bundle adjustment layer as the main component. In Sec. 6.3, the approach is realized as part of a deep visual SLAM system.

6.2.1 Deep learning and bundle adjustment

The self-calibration approach introduced in the following builds upon recent research on integrating classical bundle adjustment (Sec. 2.4.2) into a deep learning pipeline (Sec. A.1). One of the first models implementing this idea is called BA-Net and was developed for camera pose estimation [Tan18]. Using a deep neural network, the model extracts feature maps, and imposes multi-view geometry by formulating a feature-metric cost function that is optimized at inference time. Due to the similarity to classical bundle adjustment, this optimization is referred to as the bundle adjustment layer [Tan18]. During training, backpropagation is performed through the bundle adjustment layer, so that the neural network learns to provide feature maps specifically for the subsequent optimization. One of the key ideas introduced in [Tan18] is to formulate the Levenberg Marquardt algorithm used during optimization (Sec. 2.2.2) in a differentiable manner so that the loss can be backpropagated through the optimization steps during training.

Several works built upon this idea and proposed different variants: [Wei20] proposes to optimize not only poses, but also depth during bundle adjustment, [Tee18] proposes to define the cost function in terms of a reprojection error rather than a feature-metric error, and [Gu21] proposes to use a recurrent network architecture. Furthermore, [Tee18] replaces the single forward-pass through a deep neural network by an iterative approach that alternates between a deep neural network and a motion-only bundle adjustment. Finally, the idea was extended to visual SLAM, where [Tee21] proposes a combination of a recurrent network and a bundle adjustment layer within a real-time SLAM system.

Overall, research has shown that models that integrate bundle adjustment into a deep learning pipeline achieve high accuracy on pose and depth estimation, and generalize comparatively well to unseen data [Tee21]. In the following, this line of research will be extended to the problem of camera self-calibration.

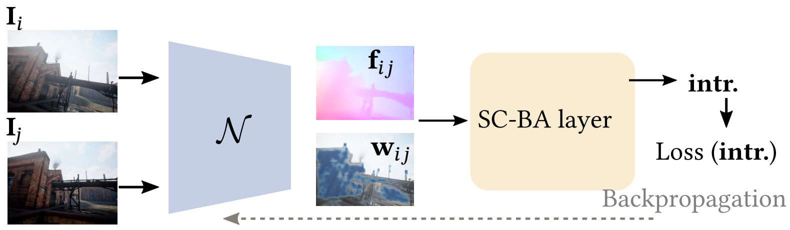


Figure 6.3: Simplified schematic depiction of the proposed self-calibration approach. Its main components are a deep neural network $\mathcal{N}$ and the self-calibrating bundle adjustment (SC-BA) layer, and the model is trained end-to-end. The network predicts dense correspondences, here visualized as optical flow $\mathbf{f}_{ij}$, and confidence weights $\mathbf{w}_{ij}$ for each correspondence. The SC-BA layer uses these predictions to optimize the intrinsics through self-calibrating bundle adjustment. See also Fig. 6.2 for comparison with existing self-calibration approaches. Figure extracted from [Hag23].

6.2.2 Model overview

Following [Hag23], this section introduces the proposed self-calibration approach conceptually based on Fig. 6.3.

Notation We denote a sequence of images by $\mathcal{I} = \{\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3} | i = 1,...N_\mathcal{I}\}$ and a set of image pairs with overlapping field-of-view by $\mathcal{P} = \{(\mathbf{I}_i, \mathbf{I}_j) | \mathbf{I}_i,\mathbf{I}_j \in \mathcal{I} \wedge \mathbf{I}_i,\mathbf{I}_j \text{ overlap}\}$. For each image pair, the grid of all pixel coordinates in source view $\mathbf{I}_i$ is denoted by $\mathbf{u}_i \in \mathbb{R}^{H\times W\times 2}$, and the corresponding points in the target view $\mathbf{I}_j$ are denoted by $\mathbf{u}_{ij} \in \mathbb{R}^{H\times W\times 2}$ where the index highlights the dependence on the source view $\mathbf{I}_i$[1]. The pixel-wise depth of an image $\mathbf{I}_i$ is denoted by $\mathbf{z}_i \in \mathbb{R}^{H\times W}$.

The poses of all images in $\mathcal{I}$ are defined w.r.t. the first image of the sequence and denoted by $\{\mathbf{G}_i\}_{i=1}^{N_\mathcal{I}}$, $\mathbf{G}_i \in SE(3)$. The relative pose between two images is given by $\mathbf{G}_{ij} = \mathbf{G}_j\mathbf{G}_i^{-1}$. As described in Sec. 2.3.1, a 3D pose transformation of points $\mathbf{x} \in \mathbb{R}^3$ in Cartesian coordinates is denoted by $\mathbf{G} * \mathbf{x} := \mathbf{R}\mathbf{x} + \mathbf{t}$ where $\mathbf{R} \in SO(3)$ denotes the rotation and $\mathbf{t} \in \mathbb{R}^3$ the translation of $\mathbf{G} \in SE(3)$.

[1] Note that for this chapter, we re-assign the variable $\mathbf{u}$ from denoting a single image point to denoting a grid of image points.

Correspondence search The input to the model is a set of image pairs $\mathcal{P}$ with overlapping field-of-view (Fig. 6.3). First, each image pair $(\mathbf{I}_i, \mathbf{I}_j)$ is passed through a deep neural network $\mathcal{N}$, where the architecture of $\mathcal{N}$ is designed such that it enables predicting *dense correspondences* $\mathbf{u}^*_{ij}$ between the images $\mathbf{I}_i$ and $\mathbf{I}_j$ and *confidence weights* $\mathbf{w}_{ij}$ associated with each correspondence[1]. The dense correspondences $\mathbf{u}^*_{ij}$ are defined by the image coordinates in image $\mathbf{I}_j$ that correspond to the grid of pixels $\mathbf{u}_i$ in image $\mathbf{I}_i$, and will be referred to as the *measured correspondences* in the following. The confidence weights $\mathbf{w}_{ij}$ are associated with each correspondence and determine how much each correspondence should contribute to the subsequent bundle adjustment (Fig. 6.3).

Self-calibrating bundle adjustment The central building block of the model is the self-calibrating bundle adjustment (SC-BA) layer (Fig. 6.3). Based on the measured correspondences $\mathbf{u}^*_{ij}$ and confidence weights $\mathbf{w}_{ij}$ of all image pairs $\mathcal{P}$, the SC-BA layer optimizes the camera intrinsics by performing differentiable self-calibrating bundle adjustment (see also Sec. 2.4.2).

Specifically, the intrinsics $\boldsymbol{\theta}$ are estimated along with camera poses $\mathbf{G}$ and depths $\mathbf{z}$ by minimizing the sum of weighted reprojection errors across all image pairs $\mathcal{P}$ [Hag23]:

$$E(\mathbf{G}, \mathbf{z}, \boldsymbol{\theta}) = \sum_{(i,j)\in\mathcal{P}} || \underbrace{\mathbf{u}^*_{ij} - \pi_C(\mathbf{G}_{ij} * \pi_C^{-1}(\mathbf{u}_i, \mathbf{z}_i, \boldsymbol{\theta}), \boldsymbol{\theta})}_{\mathbf{r}_{ij}} ||^2_{\boldsymbol{\Sigma}_{ij}}, \tag{6.1}$$

where $\boldsymbol{\Sigma}_{ij} = \operatorname{diag}(\mathbf{w}_{ij})^{-1}$ contains the confidence weights predicted by $\mathcal{N}$, $\mathbf{G} = (\mathbf{G}_1, ...\mathbf{G}_{N_\mathcal{I}})^\top$ contains all pose parameters, and $\mathbf{z} = (\mathbf{z}_1, ...\mathbf{z}_{N_\mathcal{I}})^\top$ contains the depth parameters of all images in $\mathcal{I}$. The estimates of intrinsics, poses, and depth are thus given by

$$(\widehat{\mathbf{G}}, \hat{\boldsymbol{\theta}}, \hat{\mathbf{z}}) = \operatorname*{arg\,min}_{\mathbf{G},\mathbf{z},\boldsymbol{\theta}} E(\mathbf{G}, \mathbf{z}, \boldsymbol{\theta}). \tag{6.2}$$

[1] As this task can be achieved based on different network architectures [Sar20, Sun18, Tee20, Sun21, Tee21], we here stick to an abstract description of $\mathcal{N}$.

This formulation of self-calibrating bundle adjustment differs from formulation (2.21) not only in that it is dense rather than sparse, but also in that it constrains the 3D points to the viewing rays of the respective source view $\mathbf{I}_i$. Furthermore, while the sparse formulation (2.21) globally optimizes the coordinates of 3D points across all images, formulation (6.1) separately optimizes pixel-wise depth for each keyframe $\mathbf{I}_i$, based on all image pairs containing $\mathbf{I}_i$ as source view (see Fig. 6.4 (b)). The consistency of depths across keyframes is therefore only imposed implicitly, through the consistency of poses and intrinsics across image pairs (see also discussion 6.5).

Training To train the deep neural network $\mathcal{N}$ to predict correspondences $\mathbf{u}_{ij}^*$ and confidence weights $\mathbf{w}_{ij}$ that are optimal for self-calibration, the model is trained end-to-end. To this end, the loss function $\mathcal{L}$ quantifies, among other components, the deviation of the estimated intrinsics $\hat{\boldsymbol{\theta}}$ from the true intrinsics $\bar{\boldsymbol{\theta}}$ of the training data. A specific network architecture $\mathcal{N}$ and the associated training will be introduced in Sec. 6.3. In the following, we first describe how to perform the optimization of intrinsics (6.2) in a differentiable and efficient manner, so that it can be performed within an end-to-end deep learning pipeline.

6.2.3 Gauss-Newton update including intrinsics

The structure of the optimization problem (6.2) is shown in Fig. 6.4. As (6.2) is a non-linear least squares problem, it can be solved iteratively using the Gauss-Newton algorithm (cf. Sec. 2.2.2),

$$\mathbf{J}^\top \mathbf{W} \mathbf{J} \boldsymbol{\Delta\beta} = -\mathbf{J}^\top \mathbf{W} \mathbf{r}, \tag{6.3}$$

where $\boldsymbol{\Delta\beta} = (\boldsymbol{\Delta}\mathbf{G}, \boldsymbol{\Delta\theta}, \boldsymbol{\Delta}\mathbf{z})^\top$ contains the updates of camera poses, intrinsics and pixel-wise depths, $\mathbf{r}$ is a vector containing all residuals of the cost function (6.1) and $\mathbf{J}$ is the Jacobian of the residuals w.r.t. all parameters. $\mathbf{W}$ is the weight matrix, defined by the confidence weights via $\mathbf{W} = \operatorname{diag}(\mathbf{w}_{ij})$.

To solve for the parameter updates, the block structure of the approximated Hessian $\mathbf{H} = \mathbf{J}^\top \mathbf{W} \mathbf{J}$ can be exploited [Hag23, Tee21]. Let $\tilde{\mathbf{r}} = -\mathbf{J}^\top \mathbf{W} \mathbf{r}$, then

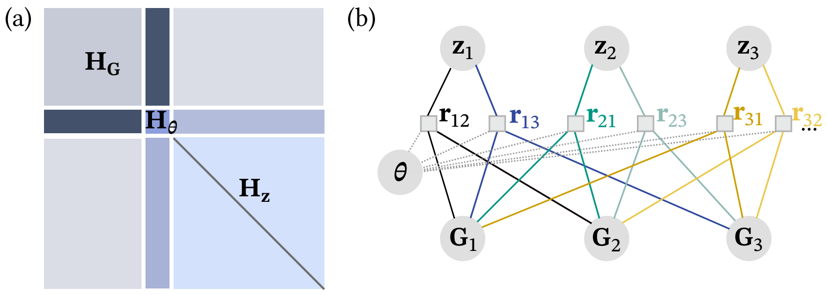


Figure 6.4: Structure of the optimization problem in the SC-BA layer. (a) Block structure of the approximated Hessian. (b) Factor graph representation of the optimization problem. The nodes represent the parameters to be estimated, including poses $\mathbf{G}$, depth $\mathbf{z}$ and intrinsics θ. The factors $\mathbf{r}_{ij}$ represent the error terms associated with each image pair $(\mathbf{I}_i, \mathbf{I}_j)$. Figure (b) extracted from [Hag23].

we can write the Gauss-Newton step as

$$\begin{bmatrix} \mathbf{H}_\mathbf{G} & \mathbf{H}_{\mathbf{G},\theta} & \mathbf{H}_{\mathbf{G},\mathbf{z}} \\ \mathbf{H}_{\mathbf{G},\theta}{}^\top & \mathbf{H}_\theta & \mathbf{H}_{\theta,\mathbf{z}} \\ \mathbf{H}_{\mathbf{G},\mathbf{z}}{}^\top & \mathbf{H}_{\theta,\mathbf{z}}{}^\top & \mathbf{H}_\mathbf{z} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{G} \\ \Delta\theta \\ \Delta\mathbf{z} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{r}}_\mathbf{G} \\ \tilde{\mathbf{r}}_\theta \\ \tilde{\mathbf{r}}_\mathbf{z} \end{bmatrix}, \tag{6.4}$$

where the indices of the Hessian blocks indicate their associated parameters (see also Fig. 6.4).

The depth block $\mathbf{H}_\mathbf{z}$ has a diagonal structure. This property can be exploited during matrix inversion, as diagonal matrices can be inverted efficiently through element-wise inversion of the diagonal elements. We therefore combine pose parameters and intrinsics by defining $\Delta\boldsymbol{\beta}_{\mathbf{G},\theta} := (\Delta\mathbf{G}, \Delta\theta)^\top$, and $\tilde{\mathbf{r}}_{\mathbf{G},\theta} := (\tilde{\mathbf{r}}_\mathbf{G}, \tilde{\mathbf{r}}_\theta)^\top$, such that the associated Hessian blocks form a joint block (Fig. 6.4).

The Gauss-Newton update (6.4) can then be rewritten as [Hag23]:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{H}_\mathbf{z} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\beta}_{\mathbf{G},\theta} \\ \Delta\mathbf{z} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{r}}_{\mathbf{G},\theta} \\ \tilde{\mathbf{r}}_\mathbf{z} \end{bmatrix}. \tag{6.5}$$

Such a block-wise equation can be solved using the Schur complement [Boy04, p.672-673]. The solution is given by [Hag23, Tee21]:

$$\begin{aligned}\boldsymbol{\Delta\beta}_{\mathbf{G},\theta} &= [\mathbf{A} - \mathbf{B}{\mathbf{H}_\mathbf{z}}^{-1}\mathbf{B}^\top]^{-1}(\tilde{\mathbf{r}}_{\mathbf{G},\theta} - \mathbf{B}{\mathbf{H}_\mathbf{z}}^{-1}\tilde{\mathbf{r}}_\mathbf{z}),\\ \boldsymbol{\Delta}\mathbf{z} &= {\mathbf{H}_\mathbf{z}}^{-1}(\tilde{\mathbf{r}}_\mathbf{z} - \mathbf{B}^\top\boldsymbol{\Delta\beta}_{\mathbf{G},\theta}),\end{aligned} \tag{6.6}$$

and the updates for poses and intrinsics can be extracted from $\boldsymbol{\Delta\beta}_{\mathbf{G},\theta}$. To improve convergence, the Gauss-Newton update (6.3) can further be augmented with a damping term (see Levenberg-Marquardt optimization, Sec. 2.2.2).

For the optimization to be differentiable, the number of bundle adjustment steps must be specified in advance instead of using convergence criteria, as if-else statements are not differentiable [Tan18, Tee21]. Furthermore, the common heuristics for choosing the damping factor λ in the Levenberg-Marquardt optimization (Sec. 2.2.2) cannot be used, as they contain if-else statements. Instead, they must be replaced with pre-defined or learned values, or implemented as an additional output of the neural network [Tan18, Tee21].

6.3 Deep self-calibrating SLAM system

In the following, we introduce a specific realization of the self-calibration approach described above. In the literature, there exists a variety of possible network architectures to predict correspondences between pairs of images [Sar20, Sun18, Tee20, Sun21, Tee21]. As shown in [Hag23], we build upon the architecture proposed by DROID-SLAM [Tee21], as it was shown to achieve high accuracy on pose estimation and to generalize well to unseen data.

DROID-SLAM uses a "Differentiable Recurrent Optimization-Inspired Design (DROID)" [Tee21]. As described in (Fig. 6.3), it uses a deep neural network to find dense correspondences and confidence weights between pairs of images. Yet, rather than directly predicting the correspondences in a single forward

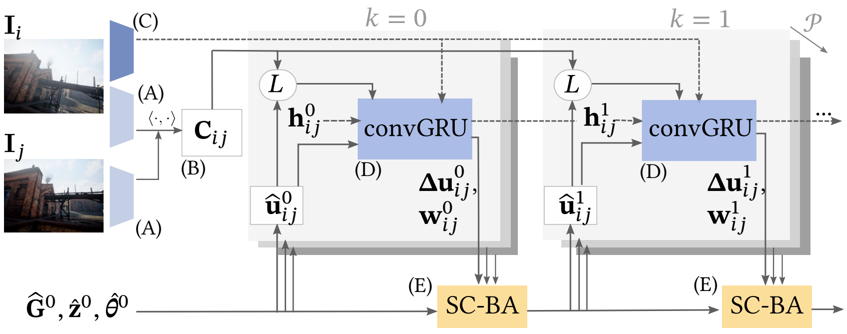

$\mathbf{I}_i$
$\mathbf{I}_j$
(C)
(A)
(A)
$\langle\cdot,\cdot\rangle$
$\mathbf{C}_{ij}$
(B)
$k=0$
$k=1$
$\mathcal{P}$
L
$\mathbf{h}^0_{ij}$
convGRU
(D)
$\hat{\mathbf{u}}^0_{ij}$
$\Delta\mathbf{u}^0_{ij}$, $\mathbf{w}^0_{ij}$
$\mathbf{h}^1_{ij}$
$\hat{\mathbf{u}}^1_{ij}$
$\Delta\mathbf{u}^1_{ij}$, $\mathbf{w}^1_{ij}$
(E)
SC-BA
$\hat{\mathbf{G}}^0, \hat{\mathbf{z}}^0, \hat{\theta}^0$
...

Figure 6.5: Integration of the self-calibrating bundle adjustment (SC-BA) layer into DROID-SLAM [Tee21]. Visualized are two unrolled iterations k. For each image pair $(\mathbf{I}_i, \mathbf{I}_j)$, the model predicts flow revisions $\Delta\mathbf{u}^k_{ij}$ and confidence weights $\mathbf{w}^k_{ij}$. Based on these predictions, the SC-BA layer updates the intrinsics θ, along with poses $\mathbf{G}$ and depth $\mathbf{z}$. A detailed depiction of the individual modules can be found in App. Fig. C.1. Figure extracted from [Hag23] and inspired by Fig. 2 in [Tee21].

pass (Fig. 6.3), it iteratively predicts updates on initially guessed correspondences, using a recurrent network architecture. At test time, DROID-SLAM is implemented as a SLAM system (Sec. A.6).

In the following, the underlying network architecture and our modifications and training for self-calibration will be described, as adopted from [Hag23]. Although the network architecture itself is not a contribution of this thesis, we summarize its relevant parts, as they contribute to an understanding of the system as a whole.

6.3.1 Architecture

An overview of the architecture is visualized in Fig. 6.5, and all parts will be described based on this overview.

Feature extraction (A) In the first step, each image $\mathbf{I}$ is passed through a *feature extraction module*, a CNN that extracts feature maps $\mathcal{F}_{\text{feat}} \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times128}$ with $\tilde{H} = H/8$ and $\tilde{W} = W/8$. In addition, the source view of each image pair is passed through the *context module* **(C)**, a second CNN that gives additional feature maps $\mathcal{F}_{\text{con}} \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times 128}$.

Correlation layer (B) For each image pair $(\mathbf{I}_i, \mathbf{I}_j)$, a four-dimensional *correlation volume* $\mathbf{C}_{ij} \in \mathbb{R}^{\tilde{W}\times\tilde{H}\times\tilde{W}\times\tilde{H}}$ is constructed which contains the inner product $\langle\cdot,\cdot\rangle$ between all feature vectors in feature map $\mathcal{F}_{\text{feat},i}$ of image $\mathbf{I}_i$, and $\mathcal{F}_{\text{feat},j}$ of image $\mathbf{I}_j$, respectively. The rationale behind this layer is that corresponding points should have similar feature vectors, thus exhibiting high correlation. Therefore, the pixel-wise correlations can provide cues for finding correspondences.

Iterative updates (D) ↔ (E) Given an initial guess[1] for poses $\widehat{\mathbf{G}}^0$, depth $\hat{\mathbf{z}}^0$ and intrinsics $\hat{\theta}^0$, the model iteratively updates all parameters. In each update iteration k, the model first predicts flow revisions **(D)**, in order to bring the estimated correspondences closer to the true correspondences. Based on the updated correspondences, the self-calibrating bundle adjustment **(E)** updates all parameters. In the following, both, **(D)** and **(E)** are described in further detail.

Prediction of flow revisions (D) In each iteration k, and for every image pair $(\mathbf{I}_i, \mathbf{I}_j) \in \mathcal{P}$ the current estimates for the relative pose $\widehat{\mathbf{G}}_{ij}^k = \widehat{\mathbf{G}}_j^k(\widehat{\mathbf{G}}_i^k)^{-1}$, the intrinsics $\hat{\theta}^k$ and depth $\hat{\mathbf{z}}_i^k$ are plugged into the multi-view constraint (2.20), to determine *predicted correspondences*,

$$\hat{\mathbf{u}}_{ij}^k = \pi_C(\widehat{\mathbf{G}}_{ij}^k * \pi_C^{-1}(\mathbf{u}_i, \hat{\mathbf{z}}_i^k, \hat{\theta}^k), \hat{\theta}^k). \tag{6.7}$$

The goal of the convolutional gated recurrent unit (convGRU) is to bring these predicted correspondences closer to the true, unknown correspondences. The convGRU is a recurrent neural network[2]. The core of the convGRU is its *hidden state* $\mathbf{h}_{ij}^k \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times 128}$. It is updated in every iteration k, such that it contains not only information on the current input but also information on the previous iterations.

[1] An initial guess for intrinsics $\hat{\theta}^0$, poses $\widehat{\mathbf{G}}^0$ and depth $\hat{\mathbf{z}}^0$ is assumed to be given but may be far from the true values.

[2] For details on the convGRU, see App. Sec. A.1 and C.1.

The inputs in each iteration k are the current estimate for the optical flow $\hat{\mathbf{f}}_{ij}^{k} := \hat{\mathbf{u}}_{ij}^{k} - \mathbf{u}_i$, the context features **(C)**, and the correlation features[1] **(B)** for each image pair $(\mathbf{I}_i, \mathbf{I}_j)$. Based on these inputs, the hidden state is updated, giving $\mathbf{h}_{ij}^{k+1}$. By mapping $\mathbf{h}_{ij}^{k+1}$ through two additional sets of convolutional layers, two outputs are generated for each image pair [Tee21, Hag23]:

- The *flow revision* $\mathbf{\Delta u}_{ij}^{k} \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times 2}$ which serves as a correction term that brings the predicted correspondences $\hat{\mathbf{u}}_{ij}^{k}$ closer to the true correspondences, and
- *confidence weights* $\mathbf{w}_{ij}^{k} \in \mathbb{R}_{+}^{\tilde{H}\times\tilde{W}\times 2}$ which quantify how certain the model is about each revision term in $\mathbf{\Delta u}_{ij}^{k}$.

Adding the flow revision $\mathbf{\Delta u}_{ij}^{k}$ to the predicted correspondences $\hat{\mathbf{u}}_{ij}^{k}$ gives the *measured correspondences*,

$$\mathbf{u}_{ij}^{*k} = \hat{\mathbf{u}}_{ij}^{k} + \mathbf{\Delta u}_{ij}^{k}, \tag{6.8}$$

which will be used in the subsequent self-calibrating bundle adjustment. Fig. 6.6 shows exemplary confidence weights and optical flow computed based on the output of the convGRU.

Self-calibrating bundle adjustment (E) Following Sec. 6.2, the SC-BA layer updates the intrinsics $\hat{\theta}^k$, poses $\hat{\mathbf{G}}^k$, and depths $\hat{\mathbf{z}}^k$ of all image pairs $\mathcal{P}$ through minimization of the reprojection error. In each iteration k, the measured correspondences $\mathbf{u}_{ij}^{*k}$ (6.8) of all image pairs in $\mathcal{P}$ are substituted in the reprojection error (6.1), giving

$$E^k(\mathbf{G}, \mathbf{z}, \theta) = \sum_{(i,j)\in\mathcal{P}} ||\mathbf{u}_{ij}^{*k} - \pi_C(\mathbf{G}_{ij} * \pi_C^{-1}(\mathbf{u}_i, z_i, \theta), \theta)||^2_{\mathbf{\Sigma}_{ij}^{k}}, \tag{6.9}$$

where $\mathbf{\Sigma}_{ij}^{k} = \operatorname{diag}\,(\mathbf{w}_{ij}^{k})^{-1}$ weights the residuals based on the confidence weights.

[1] The correlation features are obtained via the lookup operation L, which extracts only the correlations from $\mathbf{C}_{ij}$ that are in a neighborhood of the currently predicted correspondences $\hat{\mathbf{u}}_{ij}^{k}$.

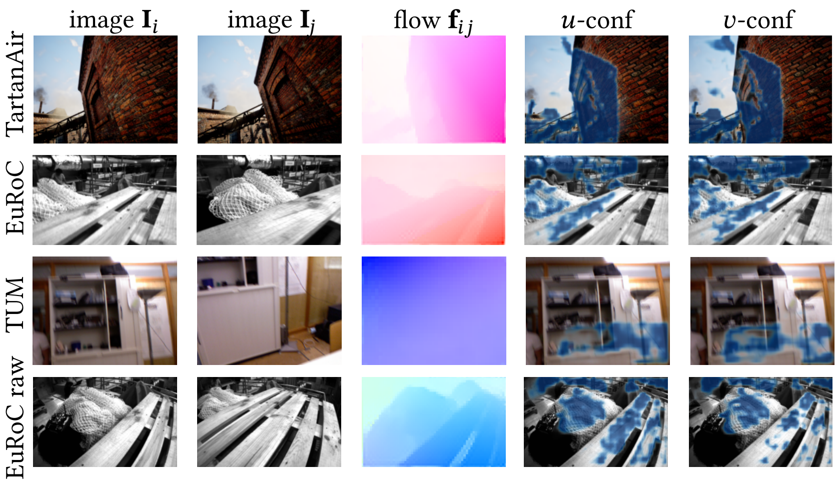


Figure 6.6: Examples of inferred flow $\mathbf{f}_{ij} = \mathbf{u}_{ij}^{*k} - \mathbf{u}_i$ and confidence weights $\mathbf{w}_{ij}^{k}$ in u- and v-direction. For flow visualization, we use the color scheme proposed in [Bak11]. For visualization of confidence weights, the transparency of the blue overlay decreases with increasing weight, using a fixed linear scaling in the range $[0, 1]$. Exemplary images are from the respective datasets [Wan20, Bur16, Stu12b].

Following Sec. 6.2, intrinsics, poses and depths are updated by minimizing the reprojection error (6.9) through differentiable non-linear least squares optimization. To improve the convergence and robustness of the optimization, a Levenberg Marquardt augmentation (Sec. 2.2.2) is used. We employ the same augmentation as [Tee21] but expand the damping of pose updates to the intrinsics. Specifically, the block-wise Gauss-Newton update derived in (6.5) is augmented to

$$\begin{bmatrix} \mathbf{A} + \lambda_{\mathbf{G},\theta}\mathbb{1} + \gamma \operatorname{diag}(\mathbf{A}) & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{H}_\mathbf{z} + \operatorname{diag}(\boldsymbol{\lambda}_z) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\beta}_{\mathbf{G},\theta} \\ \Delta\mathbf{z} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{r}}_{\mathbf{G},\theta} \\ \tilde{\mathbf{r}}_\mathbf{z} \end{bmatrix}, \tag{6.10}$$

where $\lambda_{\mathbf{G},\theta} = 0.1$ is a pre-defined damping factor for poses and intrinsics and $\gamma = 10^{-4}$ introduces a second damping term that depends on the diagonal elements of $\mathbf{A}$, here denoted by $\operatorname{diag}(\mathbf{A})$. The damping factors for the depth block, $\boldsymbol{\lambda}_\mathbf{z} \in \mathbb{R}_+^{\tilde{H}\times\tilde{W}\times N_\mathcal{J}}$, are obtained as an additional output of the convGRU, and diag($\boldsymbol{\lambda}_z$) here denotes a square matrix containing the elements of $\boldsymbol{\lambda}_\mathbf{z}$ as diagonal elements [Tee21].

The solution of (6.10) is obtained via the Schur complement as shown in (6.6), and a fixed number of $n_G = 2$ Gauss-Newton steps is performed in every iteration k, giving the updated estimates $\hat{\mathbf{G}}^{k+1}, \hat{\mathbf{z}}^{k+1}$ and $\hat{\theta}^{k+1}$.

6.3.2 Training

Following Sec. 6.2, the model is trained end-to-end such that the neural network learns to predict flow revisions and confidence weights specifically for self-calibrating bundle adjustment. We follow the supervised training setup proposed in [Tee21], using the synthetic dataset TartanAir [Wan20]: During training, groups of $N_\mathcal{J} = 7$ images are passed through the model, and in each forward pass, a fixed number of $n_k = 14$ update iterations is performed. To fix the gauge and the scale, the first two poses are fixed to the ground-truth poses.

In [Tee21], which focuses on camera pose estimation, a combination of a *pose loss*, *flow loss* and *residual loss* is used: The pose loss is defined as the average L_2-distance of the estimated poses $\hat{\mathbf{G}}^{k+1}$ from the ground truth poses $\bar{\mathbf{G}}$. The flow loss $\mathcal{L}_{\text{flow}}$ is defined by the average L_2-distance between the ground-truth optical flow field $\bar{\mathbf{f}}_{ij}$ and the estimated flow field $\hat{\mathbf{f}}_{ij}^{k+1} = \hat{\mathbf{u}}_{ij}^{k+1} - \mathbf{u}_i$, induced by the updated estimate of poses, depth, and intrinsics (6.7). Finally, the residual loss $\mathcal{L}_{\text{res}}$ penalizes the reprojection error, defined by the average absolute deviation of estimated correspondences $\hat{\mathbf{u}}_{ij}^{k+1}$ after the parameter update, from the measured correspondences $\mathbf{u}_{ij}^{*k}$ [Tee21].

To specifically train for intrinsic self-calibration, we additionally introduce an *intrinsics loss* [Hag23],

$$\mathcal{L}_\theta = \sum_{k=0}^{n_k - 1} \omega^{n_k - k} \|\hat{\theta}^{k+1} - \bar{\theta}\|_1, \tag{6.11}$$

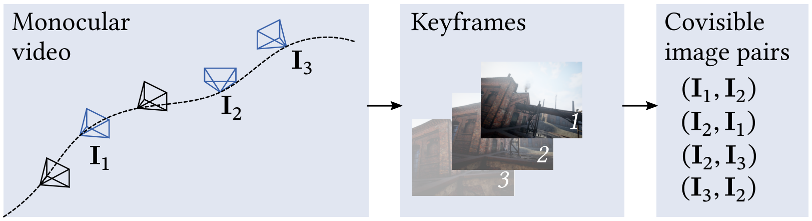

Monocular video
$\mathbf{I}_1$
$\mathbf{I}_2$
$\mathbf{I}_3$
Keyframes
1
2
3
Covisible image pairs
$(\mathbf{I}_1, \mathbf{I}_2)$
$(\mathbf{I}_2, \mathbf{I}_1)$
$(\mathbf{I}_2, \mathbf{I}_3)$
$(\mathbf{I}_3, \mathbf{I}_2)$

Figure 6.7: Depiction of keyframe selection and covisible image pair generation. The input is a monocular video with unknown intrinsics. The system selects a subset of all images as keyframes, and connects keyframes with overlapping field of view (covisible images) to form pairs. This is implemented in terms of a frame graph whose edges connect covisible keyframes.

where $\bar{\theta}$ denotes the ground-truth intrinsics from the synthetic dataset, $||\cdot||_1$ is the L_1-norm, and $\omega = 0.9$ is a weighting factor resulting in the loss contribution to increase with k, i.e. deviations after a larger number of update iterations are penalized more strongly. Furthermore, we expose the model to calibration errors during training by perturbing the initial intrinsics with uniformly distributed random errors $\delta \in [-5\%, 5\%]$. We train for 250000 iterations on two Nvidia Titan RTX 24 GB GPUs with a learning rate of 2.5×10^{-4}, a batch size of one per GPU, and using the Adam optimizer [Kin14]. For implementation, we use the PyTorch deep learning library [Pas19] and training takes ~12 days. In App. Sec. D.3, we further evaluate the impact of different training setups.

6.3.3 Inference

Inference follows the original DROID-SLAM system [Tee21], but replaces the original update operator with the self-calibration model (Fig. 6.5).

Similar to classical visual SLAM systems (e.g. ORB-SLAM [Mur15, Cam21]), DROID-SLAM has a *frontend* that operates on a monocular image stream and selects certain images as *keyframes*. It builds a *frame graph* that connects covisible keyframes (Fig. 6.7) and the update operation (Fig. 6.5) is applied on the current frame graph whenever images are added or removed, giving iterative updates of all parameters. After all images of a sequence have been registered, the *backend* performs a global optimization across the whole sequence. To this end, the frame graph is re-built using all keyframes, with edges being added based on temporal neighborhood and distance as measured by the optical flow (cf. App. Sec. C.1.2). In the backend, the update operation (Fig. 6.5) is applied for a total of $k_{\text{max}} = 19$ iterations [Tee21]. At inference time, a single Nvidia Titan RTX 24 GB GPU is used.

Implementation details We build upon the open-source implementation of DROID-SLAM [Tee21], and use the same implementation as [Hag23]. To reduce runtime and memory consumption, the SC-BA layer is implemented as a custom cuda-kernel so that matrix operations are parallelized on a GPU. Furthermore, the sparsity of the Jacobian and the Hessian blocks is exploited by storing all matrices in a sparse format and performing only the matrix operations that contain non-zero elements. Note, however, that the matrix blocks associated with the intrinsics are dense, as the intrinsics affect all pixels and all images. In App. Sec. D.4, calibration accuracy and runtime are analyzed for different sequence lengths, showing a trade-off between calibration accuracy and the required computational resources, both of which increase with an increasing sequence length.

6.4 Evaluation

In the following, the self-calibrating SLAM system, which will be referred to as *DroidCalib*, is evaluated experimentally and compared to different self-calibration baselines.

6.4.1 Datasets

Following [Hag23], we evaluate on different public datasets that contain image sequences of a moving camera and the associated ground-truth intrinsics and trajectories. Specifically, we use the monocular datasets that are also used in [Tee21], including the TartanAir monocular test split from the CVPR 2020

Figure 6.8: Exemplary images from the evaluation datasets TartanAir [Wan20], EuRoC [Bur16] and TUM [Stu12b].

SLAM challenge [Wan20], the EuRoC dataset [Bur16], and the Freiburg-1 sequences from the TUM-RGBD dataset [Stu12b] (Fig. 6.8).

The TartanAir dataset is a purely synthetic dataset consisting of different natural and urban environments, with the camera moving diversely through the scene [Wan20]. The EuRoC dataset was captured by a camera mounted on a micro aerial vehicle, moving through different indoor environments [Bur16]. The TUM freiburg-1 sequences are indoor sequences taken with a handheld rolling shutter camera [Stu12b].

As the native camera model implemented in DROID-SLAM and DroidCalib is the pinhole model (2.1), Secs. 6.4.2-6.4.3 use undistorted images, where the undistortion is performed using the reference distortion parameters provided by the datasets. In Sec. 6.4.4, DroidCalib is extended to the unified camera model, which allows it to operate on the raw images.

Before being passed through the model, the images are downsized to 512×384 for TartanAir, 512×320 for EuRoC, and 320×240 for TUM. In the final step, the inferred intrinsics are re-scaled so that the calibration result matches the original image size. As suggested in [Tee21], every other frame is skipped in the EuRoC and TUM sequences, which reduces the runtime.

The following evaluations use different initial values for the intrinsics. As a *naive* initial guess, assuming no prior knowledge about the camera, we determine the intrinsics based on the image dimensions

$$f_x^0 = \frac{H+W}{2}, f_y^0 = \frac{H+W}{2}, c_x^0 = \frac{W}{2}, c_y^0 = \frac{H}{2}. \tag{6.12}$$

In the analyses that specifically assess the performance for different deviations in the initial intrinsics, we impose uniformly distributed random errors $\delta \in [-\Delta_\theta, \Delta_\theta]$, $\Delta_\theta \in \{0\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$ on all intrinsic parameters which are specified relative to the respective ground truth parameters.

6.4.2 Accuracy of estimated intrinsics

Following [Hag23], the accuracy of the self-calibration result is assessed by comparing the inferred intrinsics $\hat{\theta}$ with the reference intrinsics $\bar{\theta}$ provided by the datasets. To quantify the deviation in terms of a single number, the effective mapping error ME (4.10) is computed, which quantifies the difference between two sets of intrinsics in image space. The mapping error is generally computed on the original image dimensions.

Tab. 6.2 shows the self-calibration results across all datasets and sequences, obtained using naive initial values θ_0 (6.12). On TartanAir, the final deviation from the ground truth intrinsics is smallest, with a median mapping error of 0.23 pixels. On EuRoC, the median mapping error is slightly higher with 0.42 pixels. On the TUM dataset, which contains significant motion blur and rolling shutter effects, the deviation is higher than on the other datasets, with a median mapping error of 3.09 pixels (Tab. 6.2). Overall, these results show that the approach enables estimating the intrinsics without targets, but that the accuracy depends on the quality of the sequence.

To relate the self-calibration accuracy to target-based calibration, we used the checkerboard calibration images available in the EuRoC and the TUM datasets (Fig. 6.9) and performed target-based calibrations (cf. Sec. 2.3, C.2), as shown in [Hag23]. Based on the available checkerboard images, we generated calibration datasets of different sizes (20 to 100 images), using the undistorted images (for details, see App. Sec. C.2). Fig. 6.9 shows that the average mapping errors obtained with target-based calibration on these datasets are in the same order of magnitude as those obtained using DroidCalib. While it must be noted that the calibration datasets of TUM and EuRoC are not ideal, as they contain many fronto-parallel target-camera configurations (see Fig. 6.9 and

Table 6.2: Accuracy of intrinsic self-calibration on the TartanAir, EuRoC and TUM datasets. Values show estimated focal lengths f_x, f_y and principal point c_x, c_y, and the mapping error ME w.r.t. the reference intrinsics, all in units of pixels. Table extracted from [Hag23].

	$\mathbf{f_x}$	$\mathbf{f_y}$	$\mathbf{c_x}$	$\mathbf{c_y}$	**ME** (pixel)
TartanAir					
Reference	**320.0**	**320.0**	**320.0**	**240.0**	
ME000	320.2	320.3	320.0	240.1	0.12
ME001	320.2	320.6	320.2	240.4	0.20
ME002	320.6	320.7	320.5	241.0	0.34
ME003	320.9	321.0	320.4	240.3	0.43
ME004	321.8	321.5	320.7	239.9	0.79
ME005	320.5	320.4	320.3	240.0	0.24
ME006	320.8	320.6	320.5	240.5	0.35
ME007	319.4	319.3	320.2	240.3	0.30
MH000	320.2	320.2	320.3	240.0	0.09
MH001	318.7	318.4	320.4	240.4	0.67
MH002	320.2	320.1	320.2	240.2	0.11
MH003	320.3	320.8	319.8	240.1	0.26
MH004	320.5	320.4	320.2	239.4	0.23
MH005	320.0	320.0	320.4	240.2	0.08
MH006	320.1	320.0	320.4	240.7	0.13
MH007	320.4	320.2	320.2	240.1	0.17
					Median: 0.23
EuRoC					
Reference	**458.7**	**457.3**	**367.2**	**248.4**	
MH_01	457.9	457.9	368.0	249.1	0.28
MH_02	457.9	457.1	367.8	248.5	0.25
MH_03	457.9	458.2	368.0	250.2	0.34
MH_04	457.4	457.2	367.7	249.1	0.38
MH_05	458.3	457.8	367.9	248.6	0.16
V1_01	459.1	459.0	368.2	250.0	0.42
V1_02	459.4	459.4	367.9	249.7	0.49
V1_03	459.6	459.5	368.3	249.4	0.55
V2_01	459.2	459.7	368.2	249.7	0.52
V2_02	459.5	459.8	367.8	249.9	0.58
V2_03	460.2	460.1	368.2	249.5	0.74
					Median: 0.42
TUM					
Reference	**517.3**	**516.5**	**318.6**	**255.3**	
360	530.2	529.4	320.4	261.1	3.71
desk	531.7	527.5	321.0	248.4	3.70
desk2	525.6	553.6	317.4	284.0	6.93
floor	530.5	518.4	323.0	240.2	3.09
room	523.2	526.1	320.9	261.8	2.22
xyz	523.4	504.0	325.3	281.2	3.03
rpy	531.2	532.7	323.7	276.2	4.66
plant	525.4	527.3	319.5	254.3	2.58
teddy	523.1	519.6	324.1	249.3	1.50
					Median: 3.09

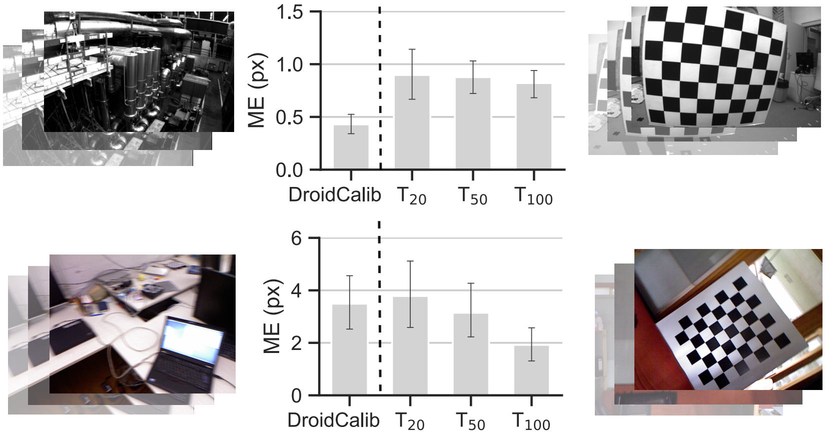


Figure 6.9: Comparison of self-calibration accuracy with target-based calibrations. T_n denotes target-based calibrations with n random images of a checkerboard target. The mapping error (ME) of DroidCalib on videos without target (left) is on the same order of magnitude as the target-based calibrations (right). The barplots show the average mapping error ME and 95% bootstrap confidence intervals. For DroidCalib, the statistics are computed across all sequences of the respective dataset, for target-based calibration they are computed across 20 random dataset samples. Figure extracted from [Hag23], exemplary images from [Bur16] and [Stu12b].

Chap. 4), the result still shows that the self-calibration can yield an accuracy in the same order of magnitude as target-based calibrations.

6.4.3 Trajectory estimation with unknown cameras

A second way to assess the self-calibration accuracy is by evaluating the accuracy of reconstructed trajectories. Following [Hag23], this section evaluates the accuracy of the reconstructed trajectories for different errors Δ_θ in the initial intrinsics and compares it to the baseline system DROID-SLAM [Tee21] that relies on known intrinsics (Fig. 6.10). Specifically, the inferred poses $\mathbf{G}$ for different errors Δ_θ are compared with the ground-truth trajectories provided by the datasets. The deviation is quantified in terms of the translation part of the average trajectory error (ATE) [Stu12a] which is computed using the *evo* toolbox [Gru17].

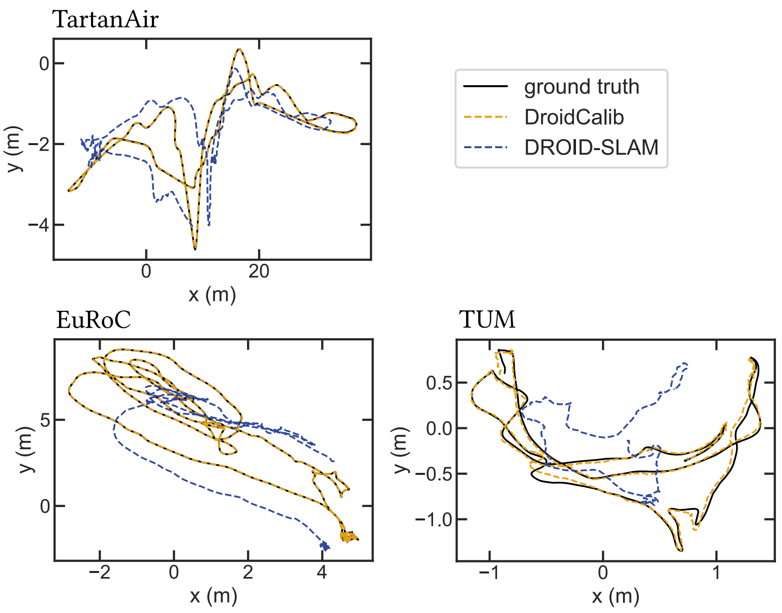


Figure 6.10: Exemplary trajectories estimated using DroidCalib and DROID-SLAM with $\Delta_\theta = 25\%$. As DroidCalib corrects the erroneous intrinsics, it infers the trajectories with high accuracy. Figure adapted from [Hag23].

The results show that if the initial intrinsics are accurate ($\Delta_\theta = 0$ %), the trajectories inferred by DroidCalib and DROID-SLAM are similarly close to the ground-truth (Fig. 6.11). As the SC-BA layer does not bring additional benefits if $\Delta_\theta = 0$, this is the expected result and it indicates that the additional degrees of freedom in the SC-BA layer do not impair the pose estimation.

For an increasing error in the initial intrinsics, the trajectories inferred by DROID-SLAM become increasingly inaccurate, which is expected as DROID-SLAM assumes these intrinsics to be accurate (Fig. 6.11). The trajectories inferred by DroidCalib, on the other hand, remain close to the ground-truth (Fig. 6.11), as the SC-BA layer corrects the erroneous intrinsics during inference. Even for $\Delta_\theta = 25$ %, the trajectories reconstructed by DroidCalib remain close to the ground-truth (Fig. 6.10). Overall, the results show that the

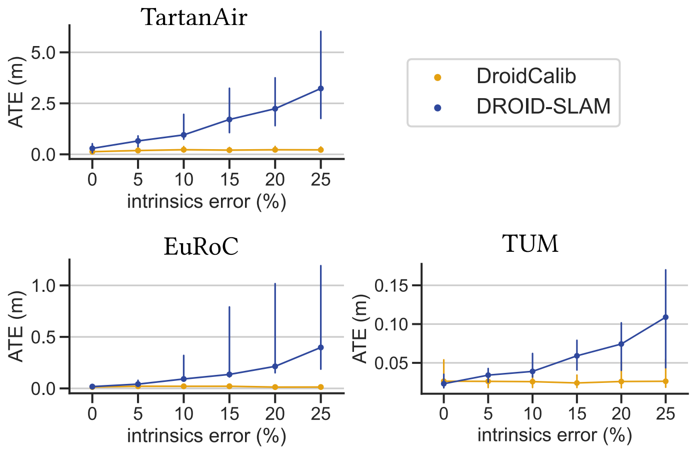


Figure 6.11: Average trajectory error (ATE) of DroidCalib and DROID-SLAM for different initial errors Δ_θ in the intrinsics. Plots show the median and 95% bootstrap confidence intervals across all sequences in the respective dataset. For each sequence, the trajectory was estimated three times, using different random errors in the initial intrinsics. Figure adapted from [Hag23].

self-calibrating bundle adjustment layer enables trajectory estimation despite inaccurate or unknown intrinsics.

6.4.4 Generalization to the unified camera model

So far, the analyses used distortion-free images and the pinhole camera model. For the self-calibration to be applicable to a wide range of cameras, this section extends it to the unified camera model (*UCM*, Sec. 2.1), as shown in [Hag23]. This is achieved by replacing the projection functions in Eqs. (6.7) and (6.1), and by computing the Jacobian in the Gauss-Newton step (6.3) accordingly. The remaining SLAM system, including the learned weights of the model, is not modified. In the following, we refer to the resulting system as *DroidCalib***.

We assess DroidCalib** on the raw images from the EuRoC dataset which exhibit significant lens distortion. Specifically, we assess the accuracy of the

Table 6.3: Trajectory estimation using the unified camera model. Applying DroidCalib** on raw images with unknown intrinsics yields similar accuracy as DROID-SLAM on undistorted images using reference intrinsics. Values show average trajectory error (ATE) in meters. The slight reduction in ATE on multiple sequences indicates that the EuRoC reference calibration might not be perfectly accurate. Table extracted from [Hag23].

	MH_01	**MH_02**	**MH_03**	**MH_04**	**MH_05**	**V1_01**	**V1_02**
DROID-SLAM	0.011	0.018	0.022	0.043	0.040	0.036	0.012
DroidCalib**	**0.008**	**0.010**	**0.020**	**0.040**	**0.037**	**0.035**	**0.010**

	V1_03	**V2_01**	**V2_02**	**V2_03**	**Median**
DROID-SLAM	**0.024**	0.016	**0.009**	0.013	0.018
DroidCalib**	0.083	**0.014**	0.010	**0.010**	**0.014**

trajectories inferred from these raw images, using naive initial intrinsics θ_0 (6.12) (Tab. 6.3).

The results show that the trajectories reconstructed by DroidCalib** on *raw* sequences with *unknown* intrinsics are as accurate as those reconstructed by DROID-SLAM on undistorted sequences with known intrinsics (on some sequences even more accurate, see Tab. 6.3). This suggests that the approach can generalize to different types of lenses without re-training.

6.4.5 Comparison with State-of-the-Art

Last, following [Hag23], we compare different state-of-the-art approaches to intrinsic self-calibration, ranging from classical Structure-from-Motion, to a deep neural network that learns the intrinsics from the training data.

Baselines As a purely classical method, we evaluate COLMAP [Sch16a], a Structure-from-Motion pipeline that allows for the refinement of camera intrinsics during reconstruction. Keypoints are extracted using SIFT [Low04] and matched across images using nearest neighbor feature matching. We use the *hloc* toolbox [Sar19] to select hyperparameters and to run the estimation. As an important hyperparameter of any SfM system is the criterion based on which images are matched with each other, we evaluate different choices, including exhaustive matching and sequential matching, in

App. Sec. C.2. As the best trade-off between runtime and accuracy, we chose to use NetVLAD [Ara16], a method that computes image descriptors and enables matching each image with its top five images with most similar descriptors.

As a recent variant of COLMAP, we evaluate COLMAP in combination with Superpoint [DeT18] and Superglue [Sar20]. Superpoint is a deep learning approach to keypoint extraction, and Superglue is a deep learning approach to feature matching. Again, we used NetVLAD [Ara16] for selecting image pairs.

As a pure deep learning-based approach, we evaluate *SelfSup-Calib*, an approach that implements the intrinsics as weights of the model which are learned during training [Fan22] (see Fig. 6.2 (c)). The original work trained jointly on all sequences of the EuRoC dataset. As our evaluation aims at per-sequence results, we instead trained on the individual sequences. Apart from this, we left all hyperparameters unchanged und used the author's open source implementation (see also App. Sec. C.2 for details).

All approaches are evaluated on TartanAir, EuRoC, TUM and the raw EuRoC images, in terms of calibration accuracy, runtime, memory consumption[1] and number of failed, non-converged runs (Tab. 6.4). For every method, we use a single run per sequence and we use naive initial values (6.12) for all intrinsics. It must be noted that DroidCalib was trained on the TartanAir training data, giving it an advantage on the TartanAir test sequences compared to the other methods (test sequences are unseen during training, but from the same domain as the training data). The sequences from EuRoC, TUM, and EuRoC raw, however, are completely novel for all methods.

Results Tab. 6.4 summarizes the results of all methods on all datasets. Across all four datasets, the intrinsics inferred by DroidCalib are closest to the ground-truth intrinsics (smallest median mapping error, Tab. 6.4). The second most accurate results are obtained with COLMAP in combination

[1] Since memory consumption is determined using *nvidia-smi*, the results for PyTorch-based implementations (SelfSup-Calib and DroidCalib) can be higher than the actual memory required, as PyTorch's caching memory allocator may reserve more memory than required [Pas19].

Table 6.4: Comparison of self-calibration baselines in terms of calibration accuracy as quantified by the median [min, max] mapping error (ME), median runtime, median peak memory consumption and number of non-converged runs, across all sequences within the respective dataset. For the raw EuRoC sequences which contain radial distortion, the COLMAP *opencv* camera model [Sch16a] (*) or the unified camera model (**) was used. COLMAP is generally used in combination with NetVLAD. The abbreviation COLMAP+SP+SG denotes COLMAP in combination with Superpoint and Superglue. Table adapted from [Hag23].

Dataset	**Method**	**ME (pixel)**		**Runtime**	**Memory**	**Failures**
TartanAir	COLMAP	1.45	[0.11, 1349.4]	**7 min**	**1 GB**	1 / 16
	COLMAP+SP+SG	0.45	[0.19, 10.8]	20 min	2 GB	0 / 16
	SelfSup-Calib**	18.3	[5.00 , 60.1]	28 min	11 GB	0 / 16
	DroidCalib	**0.23**	[0.08, 0.79]	**7 min**	11 GB	0 / 16
EuRoC	COLMAP	1.77	[0.38, 21.2]	14 min	**1 GB**	0 / 11
	COLMAP+SP+SG	0.71	[0.42, 4.11]	41 min	2 GB	0 / 11
	SelfSup-Calib**	27.6	[14.0 , 56.1]	52 min	11 GB	0 / 11
	DroidCalib	**0.42**	[0.16, 0.74]	**13 min**	12 GB	0 / 11
TUM	COLMAP	6.54	[2.53, 52.1]	**4 min**	**1 GB**	0 / 9
	COLMAP+SP+SG	4.10	[1.66, 8.09]	13 min	2 GB	2 / 9
	SelfSup-Calib**	29.7	[17.6, 44.5]	25 min	11 GB	0 / 9
	DroidCalib	**3.09**	[1.50, 6.93]	5 min	4 GB	0 / 9
EuRoC raw	COLMAP*	3.66	[2.03, 31.1]	21 min	**1 GB**	1 / 11
	COLMAP+SP+SG*	3.48	[0.66, 6.14]	62 min	2 GB	0 / 11
	SelfSup-Calib**	10.8	[1.63 , 47.9]	53 min	11 GB	0 / 11
	DroidCalib**	**0.40**	[0.31, 0.80]	**14 min**	12 GB	0 / 11

with Superpoint and Superglue, which also relies on learned features, but uses sparse keypoints without confidence weights and does not employ end-to-end training. The plain COLMAP system that uses SIFT and nearest neighbor matching yields less accurate calibration results than the variant with Superpoint and Superglue, yet, it exhibits the lowest memory consumption and comparatively short runtime. The memory consumption of deep learning-based approaches is significantly higher, with DroidCalib requiring a median peak memory of 12 GB. Finally, the self-supervised learning approach SelfSup-Calib results in the least accurate calibration results among the baselines. One reason for the low accuracy is the fact that the evaluation is performed per sequence, rather than for a batch of sequences as proposed in the original work. In App. Sec. C.2, this is analyzed in more detail.

Overall, the results suggest that the accuracy of intrinsic self-calibration benefits from the proposed combination of learned features and classical optimization, giving a reduction of the median mapping error of 49% on TartanAir, 41% on EuRoC, 25% on TUM, and 89% on EuRoC raw, compared to the respective second most accurate approaches [Hag23].

6.5 Discussion

This chapter introduces a learned self-calibration approach to estimate intrinsic camera parameters from monocular video [Hag23]. The method leverages the capabilities of deep learning while explicitly modeling projection functions and multi-view geometry. This is realized by introducing a differentiable self-calibrating bundle adjustment layer that optimizes intrinsics within an end-to-end deep learning pipeline.

Unlike plain deep learning approaches (Sec. 6.1.1), the proposed model does not require a highly diverse training dataset that contains images taken by a large variety of cameras with different intrinsics. Instead, the results show that it generalizes to different cameras by design, as the learned part of the model only performs the low-level weighted correspondence search (Tabs. 6.3, 6.4).

An important advantage compared to classical approaches that rely on handcrafted features (Sec. 6.1.1) lies in the data-driven approach to feature extraction and confidence weight prediction. Based on the statistics of the training data, the model can not only learn reliable correspondence search, but also to down-weight image areas that are unsuited for calibration, such as moving objects or unstructured sky. The observed benefit of learned features is in line with recent results on pose estimation [Lin21, Tee21, Tee22] and visual localization [Sar19, Sar21], which similarly benefit from learned features. Our results on self-calibration thereby support the notion that 3D computer vision tasks benefit from “hybrid“ approaches that combine learned features with explicit geometric modeling.

Limitations of the approach are visible on the TUM dataset, where the calibration accuracy is not as high as on the other datasets (Tab. 6.2). This can be explained by the fact that the sequences contain significant motion blur, rolling shutter effects, and less camera translation than the other sequences [Hag23]. Although the accuracy remains highest among the baselines (Tab. 6.4), a mapping error of 3 pixels is not sufficiently accurate for most applications. A second limitation lies in the the existence of *critical types of motion* [Stu97] that render intrinsic parameters unobservable [Hag23]. For instance, it can be shown that if the camera motion contains no rotation around the y-axis, the focal length f_x is not observable, and vice versa [Gor19]. Likewise, orbital motion, pure translation, and planar motion result in unobservable parameters [Stu97]. Critical types of motion are not specific to the proposed approach, but affect any intrinsic self-calibration approach that relies purely on multi-view constraints without imposing further assumptions. Accurate self-calibration therefore requires unconstrained camera motion.

Future work may expand the system to include uncertainty estimation (Chap. 4), so that challenging sequences and critical types of motion are detected during application [Hag23]. Furthermore, to better constrain the optimization (6.1), the consistency of poses and depth across keyframes could be explicitly imposed, rather than estimating pixel-wise depth for each keyframe independently, e.g. by estimating a globally consistent point cloud (see Sec. 2.4.2). Finally, although the model generalizes well to unseen data, dedicated training on challenging sequences (e.g. containing motion blur, rolling shutter effects, and little motion) may further enhance calibration accuracy on this type of data.

Overall, the proposed self-calibration opens a new way to estimate camera intrinsics from monocular video without calibration targets. It can be used to avoid the laborious target-based calibration process and to perform 3D computer vision tasks without prior knowledge about the camera.

7 Conclusion and Outlook

An accurate model of a camera's projection is essential for computer vision tasks ranging from visual SLAM, to stereo vision and neural radiance fields. As such, it builds the foundation for techniques such as automated driving and robotics, and it turns a camera into a measurement device widely used in research and industry.

This thesis introduces techniques to advance intrinsic camera calibration, including an accuracy-aware approach to target-based calibration, as well as a learned approach to self-calibration.

It is shown that calibration inaccuracies are difficult to detect based on existing techniques, as biases are typically superimposed by measurement noise (Chap. 3), and calibration datasets often do not fulfill the ideal assumptions underlying the commonly used uncertainty estimator (Chap. 4). To overcome these issues, the *bias ratio* [Hag20c, Hag22b] is introduced to disentangle bias from measurement noise, and an *approximated bootstrap approach* [Hag22b] is developed to estimate uncertainty without parametric assumptions.

The importance of assessing calibration accuracy is once again highlighted by leading to the finding that carrying calibration targets introduces dynamic deformations (Chap. 5) [Hag22c]. It is shown that these deformations can be compensated with a relatively simple deformation model which does not just improve calibration accuracy in our experiments but has meanwhile been applied to other public datasets, resulting in a significant improvement of calibration accuracy [Cvi22].

Finally, a novel approach to intrinsic camera self-calibration is introduced which enables estimating a camera's projection model based on images of an unknown environment [Hag23]. The integration of self-calibrating bundle adjustment into an end-to-end trainable deep neural network results in state-of-the-art self-calibration accuracy, and is thereby a step towards accurate estimation of projection models without targets.

Although the focus of this thesis is on monocular intrinsic calibration, all of the techniques can also be extended to stereo and multi-camera calibration. The bias ratio, the uncertainty estimation, and the deformable target model can be directly integrated into existing multi-camera calibration systems and the self-calibrating SLAM system can be extended to multi-camera systems and further modalities, as shown in [Tee21].

An interesting future direction is to extend the self-calibration approach with an estimation of bias and uncertainty, similar to target-based calibration (Chap. 3, 4). In particular, uncertainty estimation could enable the detection of critical types of motion and could thereby be used to update parameters only when they are observable (see also [May13]). This could further enhance the reliability of self-calibration in challenging scenarios.

Another interesting direction is to base the self-calibration on sparse keypoints rather than dense optical flow. Recent works have proposed learned approaches to keypoint detection [DeT18, Tan19, Gle23] and matching [Sar20, Lin23] and even sparse end-to-end trainable models for multi-view pose estimation [Roe23]. Basing self-calibration on sparse keypoints could significantly reduce memory consumption and computation time, and, by enabling operation on higher image resolution, it may further enhance self-calibration accuracy.

Overall, the findings and techniques presented in this thesis are a contribution towards accurate intrinsic calibration, both, target-based and targetless. Thereby, this thesis can contribute to the accuracy and safety of systems that rely on camera-based 3D perception.

Bibliography

[Abr98] Abraham, Steffen and Förstner, Wolfgang: “Calibration Errors in Structure from Motion”. In: *Mustererkennung 1998*. Ed. by Levi, Paul; Schanz, Michael; Ahlers, Rolf-Jürgen and May, Franz. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 117–124 (cit. on pp. 4, 27).

[Alb09] Albarelli, Andrea; Rodolà, Emanuele and Torsello, Andrea: “Robust camera calibration using inaccurate targets”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 31 (2009), pp. 376–383 (cit. on p. 63).

[Ali21] AliceVision: Photogrammetric Computer Vision Framework. meshroom. https://github.com/alicevision/meshroom/wik. Retrieved on July 29th, 2021. 2021 (cit. on p. 71).

[Ant17] Antunes, Michel; Barreto, Joao P; Aouada, Djamila and Ottersten, Bjorn: “Unsupervised vanishing point detection and camera calibration from a single manhattan image with radial distortion”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4288–4296 (cit. on pp. 3–5, 77).

[Ara16] Arandjelovic, Relja; Gronat, Petr; Torii, Akihiko; Pajdla, Tomas and Sivic, Josef: “NetVLAD: CNN architecture for weakly supervised place recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5297–5307 (cit. on pp. 99, 148).

[Bak11] Baker, Simon; Scharstein, Daniel; Lewis, JP; Roth, Stefan; Black, Michael J and Szeliski, Richard: "A database and evaluation methodology for optical flow". In: *International journal of computer vision* 92.1 (2011), pp. 1–31 (cit. on p. 88).

[Bal15] Ballas, Nicolas; Yao, Li; Pal, Chris and Courville, Aaron: "Delving deeper into convolutional networks for learning video representations". In: *arXiv preprint arXiv:1511.06432* (2015) (cit. on p. 130).

[Bar09] Barreto, Joao; Roquette, Jose; Sturm, Peter and Fonseca, Fernando: "Automatic camera calibration applied to medical endoscopy". In: *BMVC 2009-20th British Machine Vision Conference.* The British Machine Vision Association (BMVA). 2009, pp. 1–10 (cit. on p. 1).

[Bec18] Beck, Johannes and Stiller, Christoph: "Generalized B-spline Camera Model". In: *2018 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2018, pp. 2137–2142 (cit. on pp. 3, 20, 27, 29, 37–39, 49, 151).

[Bec21] Beck, Johannes: "Camera Calibration with Non-Central Local Camera Models". PhD thesis. Karlsruher Institut für Technologie (KIT), 2021. 136 pp. doi: 10.5445/IR/1000131090 (cit. on pp. 10, 28, 132, 133).

[Bla21] Blanco-Claraco, José Luis: "A tutorial on SE(3) transformation parameterizations and on-manifold optimization". In: *arXiv preprint arXiv:2103.15980* (2021) (cit. on p. 133).

[Bog18] Bogdan, Oleksandr; Eckstein, Viktor; Rameau, Francois and Bazin, Jean-Charles: "DeepCalib: A deep learning approach for automatic intrinsic calibration of wide field-of-view cameras". In: *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production.* 2018, pp. 1–10 (cit. on pp. 75, 77).

[Boy04] Boyd, Stephen; Boyd, Stephen P and Vandenberghe, Lieven: Convex optimization. Cambridge university press, 2004 (cit. on p. 84).

[Bro66] Brown, Duane C: “Decentering Distortion of Lenses”. In: *Photogrammetric Engineering* 32.3 (1966), pp. 444–462 (cit. on p. 11).

[Bur16] Burri, Michael; Nikolic, Janosch; Gohl, Pascal; Schneider, Thomas; Rehder, Joern; Omari, Sammy; Achtelik, Markus W and Siegwart, Roland: “The EuRoC micro aerial vehicle datasets”. In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1157–1163 (cit. on pp. 62, 66, 71, 88, 92, 95, 148).

[Cae20] Caesar, Holger; Bankiti, Varun; Lang, Alex H; Vora, Sourabh; Liong, Venice Erin; Xu, Qiang; Krishnan, Anush; Pan, Yu; Baldan, Giancarlo and Beijbom, Oscar: “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2020, pp. 11621–11631 (cit. on p. 4).

[Cam21] Campos, Carlos; Elvira, Richard; Rodríguez, Juan J Gómez; Montiel, José MM and Tardós, Juan D: “Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam”. In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890 (cit. on pp. 25, 75, 90, 134, 135).

[Che04] Cheong, Loong-Fah and Peh, Chin-Hwee: “Depth distortion under calibration uncertainty”. In: *Computer Vision and Image Understanding* 93.3 (2004), pp. 221–244 (cit. on pp. 4, 27).

[Che11] Cheong, Loong-Fah and Xiang, Xu: “Behaviour of SFM algorithms with erroneous calibration”. In: *Computer Vision and Image Understanding* 115.1 (2011), pp. 16–30 (cit. on pp. 4, 27).

[Che19] Chen, Yuhua; Schmid, Cordelia and Sminchisescu, Cristian: “Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 7063–7072 (cit. on p. 76).

[Cho14] Cho, Kyunghyun; Van Merriënboer, Bart; Bahdanau, Dzmitry and Bengio, Yoshua: “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014) (cit. on p. 129).

[Chu14] Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun and Bengio, Yoshua: “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014) (cit. on p. 129).

[Cip99] Cipolla, Roberto; Drummond, Tom and Robertson, Duncan P: “Camera Calibration from Vanishing Points in Image of Architectural Scenes.” In: *BMVC*. Vol. 99. 1999, pp. 382–391 (cit. on pp. 3–5, 77, 78).

[Civ09] Civera, Javier; Bueno, Diana R; Davison, Andrew J and Montiel, JMM: “Camera self-calibration for sequential bayesian structure from motion”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 403–408 (cit. on p. 75).

[Cra20] Cramariuc, Andrei; Petrov, Aleksandar; Suri, Rohit; Mittal, Mayank; Siegwart, Roland and Cadena, Cesar: “Learning camera miscalibration detection”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 4997–5003 (cit. on pp. 49, 73).

[Cvi21] Cvišić, Igor; Marković, Ivan and Petrović, Ivan: “Recalibrating the KITTI dataset camera setup for improved odometry accuracy”. In: *2021 European Conference on Mobile Robots (ECMR)*. IEEE. 2021, pp. 1–6 (cit. on pp. 3, 4, 27).

[Cvi22] Cvišić, Igor; Marković, Ivan and Petrović, Ivan: “Enhanced calibration of camera setups for high-performance visual odometry”. In: *Robotics and Autonomous Systems* 155 (2022), p. 104189 (cit. on pp. 3, 4, 27, 72, 103).

[Daa19] Daakir, M; Zhou, Y; Deseilligny, M Pierrot; Thom, C; Martin, O and Rupnik, E: “Improvement of photogrammetric accuracy by modeling and correcting the thermal effect on camera calibration”. In: *ISPRS journal of photogrammetry and remote sensing* 148 (2019), pp. 142–155 (cit. on p. 73).

[Dav97] Davison, Anthony Christopher and Hinkley, David Victor: Bootstrap methods and their application. 1. Cambridge university press, 1997 (cit. on pp. 44–46, 154).

[DeT18] DeTone, Daniel; Malisiewicz, Tomasz and Rabinovich, Andrew: “Superpoint: Self-supervised interest point detection and description”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 2018, pp. 224–236 (cit. on pp. 99, 104, 148).

[Dua71] Duane, C Brown: “Close-range camera calibration”. In: *Photogramm. Eng* 37.8 (1971), pp. 855–866 (cit. on p. 3).

[Fan22] Fang, Jiading; Vasiljevic, Igor; Guizilini, Vitor; Ambrus, Rares; Shakhnarovich, Greg; Gaidon, Adrien and Walter, Matthew R: “Self-Supervised Camera Self-Calibration from Video”. In: *2022 International Conference on Robotics and Automation (ICRA).* IEEE. 2022, pp. 8468–8475 (cit. on pp. 4, 28, 62, 74, 76, 77, 99, 149, 150).

[Far20] Farley, Kenneth A; Williford, Kenneth H; Stack, Kathryn M; Bhartia, Rohit; Chen, Al; Torre, Manuel de la; Hand, Kevin; Goreva, Yulia; Herd, Christopher DK; Hueso, Ricardo et al.: “Mars 2020 mission overview”. In: *Space Science Reviews* 216 (2020), pp. 1–41 (cit. on p. 1).

[Fau92] Faugeras, Olivier D; Luong, Q -T and Maybank, Stephen J: “Camera self-calibration: Theory and experiments”. In: *Computer Vision—ECCV’92: Second European Conference on Computer Vision Santa Margherita Ligure, Italy, May 19–22, 1992 Proceedings 2.* Springer. 1992, pp. 321–334 (cit. on pp. 3, 4, 74, 75).

[För16] Förstner, Wolfgang and Wrobel, Bernhard P: Photogrammetric computer vision. Springer, 2016 (cit. on pp. 9, 16, 132, 133).

[Fox15] Fox, John: Applied regression analysis and generalized linear models. Sage Publications, 2015 (cit. on p. 46).

[Gle23] Gleize, Pierre; Wang, Weiyao and Feiszli, Matt: “SiLK–Simple Learned Keypoints”. In: *arXiv preprint arXiv:2304.06194* (2023) (cit. on p. 104).

[Goo16] GOODFELLOW, Ian; BENGIO, Yoshua and COURVILLE, Aaron: Deep Learning. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 127, 128).

[Gor19] GORDON, Ariel; LI, Hanhan; JONSCHKOWSKI, Rico and ANGELOVA, Anelia: “Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 8977–8986 (cit. on pp. 4, 76, 77, 102).

[GOS21] GOSSOW, DAVID: “CAMERA CALIBRATION USING SYNTHETIC IMAGES”. CA2999133 C. 2021 (cit. on p. 4).

[Gru17] GRUPP, Michael: evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo. Retrieved on March 8th, 2022. 2017 (cit. on p. 95).

[Gu21] GU, Xiaodong; YUAN, Weihao; DAI, Zuozhuo; TANG, Chengzhou; ZHU, Siyu and TAN, Ping: “Dro: Deep recurrent optimizer for structure-from-motion”. In: *arXiv preprint arXiv:2103.13201* (2021) (cit. on p. 79).

[Hag20a] HAGEMANN, Annika and KNORR, Moritz: “Verfahren zum Bewerten einer Kamerakalibrierung”. DE102020211502A1, US20220086425A1, CN114189672A. Filed by Robert Bosch GmbH. 2020 (cit. on p. 5).

[Hag20b] HAGEMANN, Annika and KNORR, Moritz: “Verfahren zum Bewerten einer Kamerakalibrierung”. DE102020211507A1, US11748910B2, CN114189673A. Filed by Robert Bosch GmbH. 2020 (cit. on p. 5).

[Hag20c] HAGEMANN, Annika; KNORR, Moritz; JANSSEN, Holger and STILLER, Christoph: “Bias detection and prediction of mapping errors in camera calibration”. In: *DAGM German Conference on Pattern Recognition.* Springer. 2020, pp. 30–43 (cit. on pp. 5–8, 19, 28–32, 38, 42, 44, 49–53, 56, 59, 103, 137, 138, 140, 151).

[Hag21a] Hagemann, Annika and Knorr, Moritz: “Verfahren zur Ermittlung einer Kalibrierungsgüte eines optischen Sensors”. DE102021202993A1. Filed by Robert Bosch GmbH. 2021 (cit. on p. 5).

[Hag21b] Hagemann, Annika and Knorr, Moritz: “Verfahren zur Kalibrierung von Kamerasystemen”. DE102021210526A1. Filed by Robert Bosch GmbH. 2021 (cit. on p. 5).

[Hag22a] Hagemann, Annika and Knorr, Moritz: “Verfahren für eine Selbstkalibrierung wenigstens einer Kamera”. Aktenzeichen 102022208757.7, not yet published. Filed by Robert Bosch GmbH. 2022 (cit. on p. 5).

[Hag22b] Hagemann, Annika; Knorr, Moritz; Janssen, Holger and Stiller, Christoph: “Inferring bias and uncertainty in camera calibration”. In: *International Journal of Computer Vision* 130.1 (2022), pp. 17–32 (cit. on pp. 5–8, 19, 22, 28–38, 41–44, 46–59, 103, 137, 138, 140, 151, 152, 154, 155).

[Hag22c] Hagemann, Annika; Knorr, Moritz and Stiller, Christoph: “Modeling dynamic target deformation in camera calibration”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 1747–1755 (cit. on pp. 5, 6, 8, 19, 23, 61–68, 70–72, 103).

[Hag23] Hagemann, Annika; Knorr, Moritz and Stiller, Christoph: “Deep geometry-aware camera self-calibration from video”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 3438–3448 (cit. on pp. 5–8, 10, 24, 74, 77, 78, 80–85, 87, 89, 91, 93–98, 100–103, 140–142, 147–150, 156–159).

[Hän17] Häne, Christian; Heng, Lionel; Lee, Gim Hee; Fraundorfer, Friedrich; Furgale, Paul; Sattler, Torsten and Pollefeys, Marc: “3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection”. In: *Image and Vision Computing* 68 (2017), pp. 14–27 (cit. on pp. 1, 4).

[Har04] Hartley, Richard and Zisserman, Andrew: Multiple view geometry in computer vision. English. OCLC: 171123855. 2004 (cit. on pp. 10, 11, 18, 19, 22, 29).

[Har94] Hartley, Richard I: "Self-calibration from multiple views with a rotating camera". In: *European Conference on Computer Vision.* Springer. 1994, pp. 471–478 (cit. on p. 76).

[Hay21] Hayes, Alexander G; Corlies, P; Tate, C; Barrington, M; Bell, JF; Maki, JN; Caplinger, M; Ravine, M; Kinch, KM; Herkenhoff, K et al.: "Pre-flight calibration of the Mars 2020 Rover Mastcam Zoom (Mastcam-Z) multispectral, stereoscopic imager". In: *Space science reviews* 217 (2021), pp. 1–95 (cit. on p. 73).

[He16] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778 (cit. on p. 145).

[Hem03] Hemayed, Elsayed E: "A survey of camera self-calibration". In: *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.* IEEE. 2003, pp. 351–357 (cit. on p. 75).

[Hen13] Heng, Lionel; Li, Bo and Pollefeys, Marc: "Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE. 2013, pp. 1793–1800 (cit. on pp. 25, 75).

[Hen15a] Heng, Lionel; Furgale, Paul and Pollefeys, Marc: "Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig". In: *Journal of Field Robotics* 32.5 (2015), pp. 775–802 (cit. on pp. 3, 25, 75).

[Hen15b] Heng, Lionel; Lee, Gim Hee and Pollefeys, Marc: "Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle". In: *Autonomous robots* 39 (2015), pp. 259–277 (cit. on pp. 25, 75).

[Hoc97] Hochreiter, Sepp and Schmidhuber, Jürgen: “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 129).

[Hua13] Huang, Lei; Zhang, Qican and Asundi, Anand: “Flexible camera calibration using not-measured imperfect target”. In: *Applied optics* 52.25 (2013), pp. 6278–6286 (cit. on p. 63).

[IRR23] IRRENHAUSER, THOMAS: “METHOD FOR VALIDATING A CAMERA CALIBRATION FOR A MOVABLE ROBOT ARM BY MEANS OF A SYSTEM, COMPUTER PROGRAM PRODUCT AND SYSTEM ”. DE102021123245 A1. 2023 (cit. on p. 4).

[Jam21] James, Gareth; Witten, Daniela; Hastie, Trevor and Tibshirani, Robert: An introduction to statistical learning. Vol. 2nd ed. Springer, 2021 (cit. on p. 17).

[Jeo21] Jeong, Yoonwoo; Ahn, Seokjun; Choy, Christopher; Anandkumar, Anima; Cho, Minsu and Park, Jaesik: “Self-calibrating neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2021, pp. 5846–5854 (cit. on p. 76).

[Kei15] Keivan, Nima and Sibley, Gabe: “Online SLAM with any-time self-calibration and automatic change detection”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. 2015, pp. 5775–5782 (cit. on p. 75).

[Kin14] Kingma, Diederik P and Ba, Jimmy: “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 90).

[Lag20] Laga, Hamid; Jospin, Laurent Valentin; Boussaid, Farid and Bennamoun, Mohammed: “A survey on deep learning techniques for stereo-based depth estimation”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.4 (2020), pp. 1738–1764 (cit. on pp. 5, 127).

[Lav98] Lavest, Jean-Marc; Viala, Marc and Dhome, Michel: “Do we really need an accurate calibration pattern to achieve a reliable camera calibration?” In: *European Conference on Computer Vision*. Springer. 1998, pp. 158–174 (cit. on pp. 62, 63).

[LeC15] LeCun, Yann; Bengio, Yoshua and Hinton, Geoffrey: “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444 (cit. on pp. 127–129).

[Lee21] Lee, Jinwoo; Go, Hyunsung; Lee, Hyunjoon; Cho, Sunghyun; Sung, Minhyuk and Kim, Junho: “CTRL-C: Camera calibration TRansformer with Line-Classification”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16228–16237 (cit. on p. 77).

[Lev44] Levenberg, Kenneth: “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168 (cit. on p. 15).

[Lin20] Lin, Yukai; Larsson, Viktor; Geppert, Marcel; Kukelova, Zuzana; Pollefeys, Marc and Sattler, Torsten: “Infrastructure-based multi-camera calibration using radial projections”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 327–344 (cit. on p. 76).

[Lin21] Lindenberger, Philipp; Sarlin, Paul-Edouard; Larsson, Viktor and Pollefeys, Marc: “Pixel-Perfect Structure-from-Motion with Featuremetric Refinement”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5987–5997 (cit. on pp. 5, 101).

[Lin23] Lindenberger, Philipp; Sarlin, Paul-Edouard and Pollefeys, Marc: “LightGlue: Local Feature Matching at Light Speed”. In: *arXiv preprint arXiv:2306.13643* (2023) (cit. on p. 104).

[LIU20] LIU, PENG; CHEN, JIABO; YU, MOCHEN; YU, YANG and JIA, ZHIJIA: “BINOCULAR CAMERA CALIBRATION TOOL”. CN111192330 A. 2020 (cit. on p. 4).

[Lop19] Lopez, Manuel; Mari, Roger; Gargallo, Pau; Kuang, Yubin; Gonzalez-Jimenez, Javier and Haro, Gloria: “Deep single image camera calibration with radial distortion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 11817–11825 (cit. on pp. 77, 78).

[Low04] Lowe, David G: “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60 (2004), pp. 91–110 (cit. on pp. 98, 148).

[LU21] LU, YUESHENG and BYRNE, STEVEN V: “MULTI-CAMERA IMAGE STITCHING CALIBRATION SYSTEM ”. US10904489 B2. 2021 (cit. on p. 4).

[Luh13] Luhmann, T.; Robson, S.; Kyle, S. and Boehm, J.: Close-range Photogrammetry and 3D Imaging. De Gruyter textbook. De Gruyter, 2013 (cit. on p. 19).

[Mac22] Macario Barros, Andréa; Michel, Maugan; Moline, Yoann; Corre, Gwenolé and Carrel, Frédérick: “A comprehensive survey of visual slam algorithms”. In: *Robotics* 11.1 (2022), p. 24 (cit. on p. 135).

[Mag03] Magnan, Pierre: “Detection of visible photons in CCD and CMOS: A comparative view”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 504.1-3 (2003), pp. 199–212 (cit. on p. 9).

[Mak20] Maki, JN; Gruel, D; McKinney, C; Ravine, MA; Morales, M; Lee, D; Willson, R; Copley-Woods, D; Valvo, M; Goodsall, T et al.: “The Mars 2020 Engineering Cameras and microphone on the perseverance rover: A next-generation imaging system for Mars exploration”. In: *Space Science Reviews* 216 (2020), pp. 1–48 (cit. on pp. 1, 2, 4).

[Mar19] Marti, Enrique; De Miguel, Miguel Angel; Garcia, Fernando and Perez, Joshue: “A review of sensor technologies for perception in automated driving”. In: *IEEE Intelligent Transportation Systems Magazine* 11.4 (2019), pp. 94–108 (cit. on pp. 1, 2).

[Mar63] Marquardt, Donald W: “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441 (cit. on pp. 15, 16).

[Mat23] MathWorks: Evaluating the Accuracy of Single Camera Calibration. https://de.mathworks.com/help/vision/ug/evaluating-the-accuracy-of-single-camera-calibration.html. Retrieved on April 27st, 2023. 2023 (cit. on p. 28).

[May13] Maye, Jérôme; Furgale, Paul and Siegwart, Roland: “Self-supervised calibration for robotic systems”. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2013, pp. 473–480 (cit. on p. 104).

[May92] Maybank, Stephen J and Faugeras, Olivier D: “A theory of self-calibration of a moving camera”. In: *International journal of computer vision* 8.2 (1992), pp. 123–151 (cit. on pp. 3, 4, 75).

[Mei07] Mei, Christopher and Rives, Patrick: “Single view point omnidirectional camera calibration from planar grids”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 3945–3950 (cit. on p. 12).

[Mic22] Michel, Patrick; Küppers, Michael; Bagatin, Adriano Campo; Carry, Benoit; Charnoz, Sébastien; De Leon, Julia; Fitzsimmons, Alan; Gordo, Paulo; Green, Simon F; Hérique, Alain et al.: “The ESA Hera mission: detailed characterization of the DART impact outcome and of the binary asteroid (65803) Didymos”. In: *The Planetary Science Journal* 3.7 (2022), p. 160 (cit. on p. 1).

[Mur15] Mur-Artal, Raul; Montiel, Jose Maria Martinez and Tardos, Juan D: “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163 (cit. on pp. 2, 25, 75, 90, 135).

[NAS23] NASA: Perseverance's Three Forks Sample Depot Selfie, taken on January 22, 2023. https://mars.nasa.gov/resources/27262/perseverances-three-forks-sample-depot-selfie/. Retrieved on March 9th, 2023. 2023 (cit. on p. 1).

[Noc06] Nocedal, Jorge and Wright, Stephen J.: Numerical optimization. 2nd ed. Springer series in operations research. OCLC: ocm68629100. New York: Springer, 2006 (cit. on pp. 14–16, 131, 132, 134).

[Ope21] OpenCV: OpenCV Fisheye Camera Model. https://docs.opencv.org/master/db/d58/group__calib3d__fisheye.html. Retrieved on January 8th, 2021. 2021. url: https://docs.opencv.org/master/db/d58/group__calib3d__fisheye.html (cit. on p. 11).

[Ope23] OpenCV: OpenCV Tutorial Camera Calibrations. https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html. Retrieved on April 27st, 2023. 2023 (cit. on pp. 4, 28).

[Özg20] Özgüner, Orhan; Shkurti, Thomas; Huang, Siqi; Hao, Ran; Jackson, Russell C; Newman, Wyatt S and Çavuşoğlu, M Cenk: "Camera-robot calibration for the da vinci robotic surgery system". In: *IEEE Transactions on Automation Science and Engineering* 17.4 (2020), pp. 2154–2161 (cit. on p. 1).

[Ozo13] Ozog, Paul and Eustice, Ryan M: "On the importance of modeling camera calibration uncertainty in visual SLAM". In: *2013 IEEE International Conference on Robotics and Automation.* IEEE. 2013, pp. 3777–3784 (cit. on pp. 4, 27).

[Pas19] Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca et al.: "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. 90, 99).

[Pen19] Peng, Songyou and Sturm, Peter: "Calibration wizard: A guidance system for camera calibration based on modelling geometric and corner uncertainty". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2019, pp. 1497–1505 (cit. on pp. 3, 4, 21, 41–43, 58, 62, 66, 70, 71).

[Pen22] Peng, Rui; Wang, Rongjie; Wang, Zhenyu; Lai, Yawen and Wang, Ronggang: "Rethinking depth estimation for multi-view stereo: A unified representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pp. 8645–8654 (cit. on p. 2).

[Pxh18] Pxhere: Street scene in Japan, taken by an unknown photographer on May 10, 2018. https://pxhere.com/en/photo/1551481. Creative Commons CC0 license, retrieved on April 9th, 2023. 2018 (cit. on p. 2).

[Red16] Redmon, Joseph; Divvala, Santosh; Girshick, Ross and Farhadi, Ali: "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 779–788 (cit. on p. 127).

[Ric13] Richardson, Andrew; Strom, Johannes and Olson, Edwin: "AprilCal: Assisted and repeatable camera calibration". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE. 2013, pp. 1814–1821 (cit. on pp. 11, 41–43, 49, 56, 58, 62, 66, 67, 71).

[Roe23] Roessle, Barbara and Nießner, Matthias: "End2End Multi-View Feature Matching with Differentiable Pose Optimization". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2023, pp. 477–487 (cit. on p. 104).

[Roj18] Rojtberg, Pavel and Kuijper, Arjan: "Efficient pose selection for interactive camera calibration". In: *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR).* IEEE. 2018, pp. 31–36 (cit. on pp. 21, 41, 43, 58).

[Ron15] Ronneberger, Olaf; Fischer, Philipp and Brox, Thomas: "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18.* Springer. 2015, pp. 234–241 (cit. on p. 127).

[ROS21] ROS: ROS Tutorial MonocularCalibration. http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration. Retrieved on January 8th, 2021. 2021 (cit. on p. 4).

[Rou93] Rousseeuw, Peter J and Croux, Christophe: "Alternatives to the median absolute deviation". In: *Journal of the American Statistical association* 88.424 (1993), pp. 1273–1283 (cit. on p. 32).

[Rub19] Rubio, Francisco; Valero, Francisco and Llopis-Albert, Carlos: "A review of mobile robots: Concepts, methods, theoretical framework, and applications". In: *International Journal of Advanced Robotic Systems* 16.2 (2019), p. 1729881419839596 (cit. on p. 1).

[Sap18] Saputra, Muhamad Risqi U; Markham, Andrew and Trigoni, Niki: "Visual SLAM and structure from motion in dynamic environments: A survey". In: *ACM Computing Surveys (CSUR)* 51.2 (2018), pp. 1–36 (cit. on p. 5).

[Sar19] Sarlin, Paul-Edouard; Cadena, Cesar; Siegwart, Roland and Dymczyk, Marcin: "From coarse to fine: Robust hierarchical localization at large scale". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 12716–12725 (cit. on pp. 2, 98, 101, 148).

[Sar20] Sarlin, Paul-Edouard; DeTone, Daniel; Malisiewicz, Tomasz and Rabinovich, Andrew: "Superglue: Learning feature matching with graph neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2020, pp. 4938–4947 (cit. on pp. 5, 81, 84, 99, 104, 148).

[Sar21] Sarlin, Paul-Edouard; Unagar, Ajaykumar; Larsson, Mans; Germain, Hugo; Toft, Carl; Larsson, Viktor; Pollefeys, Marc; Lepetit, Vincent; Hammarstrand, Lars; Kahl, Fredrik et al.: "Back to the feature: Learning robust camera localization from pixels to pose". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 3247–3257 (cit. on pp. 5, 101).

[Sch16a] Schonberger, Johannes L and Frahm, Jan-Michael: "Structure-from-motion revisited". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113 (cit. on pp. 3–5, 11, 75, 77, 98, 100, 134, 148).

[Sch16b] Schönberger, Johannes Lutz; Zheng, Enliang; Pollefeys, Marc and Frahm, Jan-Michael: "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *European Conference on Computer Vision (ECCV)*. 2016 (cit. on p. 25).

[Sch20] Schops, Thomas; Larsson, Viktor; Pollefeys, Marc and Sattler, Torsten: "Why having 10,000 parameters in your camera model is better than twelve". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2535–2544 (cit. on pp. 3, 4, 20, 27, 29, 37–39, 49, 151).

[Sem16] Semeniuta, Oleksandr: "Analysis of camera calibration with respect to measurement accuracy". In: *Procedia Cirp* 41 (2016), pp. 765–770 (cit. on p. 42).

[Smi10] Smith, MJ and Cope, E: "The effects of temperature variation on single-lens-reflex digital camera calibration parameters". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 38.Part 5 (2010) (cit. on p. 73).

[Str08] Strobl, Klaus H and Hirzinger, Gerd: "More accurate camera and hand-eye calibrations with unknown grid pattern dimensions". In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 1398–1405 (cit. on pp. 27, 63).

[Str11] Strobl, Klaus H and Hirzinger, Gerd: "More accurate pinhole camera calibration with imperfect planar target". In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops).* IEEE. 2011, pp. 1068–1075 (cit. on pp. 22, 23, 63, 65).

[Str14] Strauss, Tobias; Ziegler, Julius and Beck, Johannes: "Calibrating multiple cameras with non-overlapping views using coded checkerboard targets". In: *17th international IEEE conference on intelligent transportation systems (ITSC).* IEEE. 2014, pp. 2623–2628 (cit. on pp. 20–22, 33, 34).

[Str15] Strauss, Tobias: "Kalibrierung von Multi-Kamera-Systemen". PhD thesis. Karlsruher Institut für Technologie (KIT), 2015 (cit. on pp. 22, 43, 50, 52, 56, 58).

[Stu12a] Sturm, Jürgen; Burgard, Wolfram and Cremers, Daniel: "Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark". In: *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS).* Vol. 13. 2012 (cit. on p. 95).

[Stu12b] Sturm, Jürgen; Engelhard, Nikolas; Endres, Felix; Burgard, Wolfram and Cremers, Daniel: "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ international conference on intelligent robots and systems.* IEEE. 2012, pp. 573–580 (cit. on pp. 88, 92, 95, 148).

[Stu97] Sturm, Peter: "Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction". In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE. 1997, pp. 1100–1105 (cit. on pp. 5, 102).

[Stu99] Sturm, Peter F and Maybank, Stephen J: "On plane-based camera calibration: A general algorithm, singularities, applications". In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149).* Vol. 1. IEEE. 1999, pp. 432–437 (cit. on p. 41).

[Sun06] Sun, Wei and Cooperstock, Jeremy R: "An empirical evaluation of factors influencing camera calibration accuracy using three publicly available techniques". In: *Machine Vision and Applications* 17.1 (2006), pp. 51–67 (cit. on p. 42).

[Sun18] Sun, Deqing; Yang, Xiaodong; Liu, Ming-Yu and Kautz, Jan: "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2018, pp. 8934–8943 (cit. on pp. 81, 84).

[Sun21] Sun, Jiaming; Shen, Zehong; Wang, Yuang; Bao, Hujun and Zhou, Xiaowei: "LoFTR: Detector-free local feature matching with transformers". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2021, pp. 8922–8931 (cit. on pp. 5, 81, 84).

[Svo96] Svoboda, Tomas and Sturm, Peter: "What can be done with a badly calibrated Camera in Ego-Motion Estimation?" In: (1996). url: https://hal.inria.fr/inria-00525701/ (visited on 05/14/2020) (cit. on pp. 4, 27).

[Tai18] Taira, Hajime; Okutomi, Masatoshi; Sattler, Torsten; Cimpoi, Mircea; Pollefeys, Marc; Sivic, Josef; Pajdla, Tomas and Torii, Akihiko: "InLoc: Indoor visual localization with dense matching and view synthesis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 7199–7209 (cit. on p. 148).

[Tak17] Taketomi, Takafumi; Uchiyama, Hideaki and Ikeda, Sei: "Visual SLAM algorithms: A survey from 2010 to 2016". In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), pp. 1–11 (cit. on pp. 25, 75, 135).

[Tan18] Tang, Chengzhou and Tan, Ping: "Ba-net: Dense bundle adjustment network". In: *arXiv preprint arXiv:1806.04807* (2018) (cit. on pp. 79, 84).

[Tan19] Tang, Jiexiong; Kim, Hanme; Guizilini, Vitor; Pillai, Sudeep and Ambrus, Rares: “Neural outlier rejection for self-supervised keypoint learning”. In: *arXiv preprint arXiv:1912.10615* (2019) (cit. on p. 104).

[Tan22] Tancik, Matthew; Casser, Vincent; Yan, Xinchen; Pradhan, Sabeek; Mildenhall, Ben; Srinivasan, Pratul P; Barron, Jonathan T and Kretzschmar, Henrik: “Block-nerf: Scalable large scene neural view synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pp. 8248–8258 (cit. on p. 2).

[Tar07] Tardif, Jean-Philippe; Sturm, Peter and Roy, Sébastien: “Plane-based self-calibration of radial distortion”. In: *2007 IEEE 11th International Conference on Computer Vision.* IEEE. 2007, pp. 1–8 (cit. on p. 76).

[Tee18] Teed, Zachary and Deng, Jia: “Deepv2d: Video to depth with differentiable structure from motion”. In: *arXiv preprint arXiv:1812.04605* (2018) (cit. on p. 79).

[Tee20] Teed, Zachary and Deng, Jia: “Raft: Recurrent all-pairs field transforms for optical flow”. In: *European conference on computer vision.* Springer. 2020, pp. 402–419 (cit. on pp. 5, 81, 84, 127, 145, 146, 158).

[Tee21] Teed, Zachary and Deng, Jia: “Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 16558–16569 (cit. on pp. 7, 79, 81, 82, 84, 85, 87–92, 95, 101, 104, 135, 145–148, 156–158).

[Tee22] Teed, Zachary; Lipson, Lahav and Deng, Jia: “Deep patch visual odometry”. In: *arXiv preprint arXiv:2208.04726* (2022) (cit. on p. 101).

[Tri00] Triggs, Bill; McLauchlan, Philip F; Hartley, Richard I and Fitzgibbon, Andrew W: “Bundle adjustment—a modern synthesis”. In: *Vision Algorithms: Theory and Practice: International*

Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings. Springer. 2000, pp. 298–372 (cit. on pp. 52, 138).

[Tri99] Triggs, Bill; McLauchlan, Philip F; Hartley, Richard I and Fitzgibbon, Andrew W: "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms.* Springer. 1999, pp. 298–372 (cit. on pp. 15, 24, 25, 75).

[Tsa87] Tsai, Roger: "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* 3.4 (1987), pp. 323–344 (cit. on pp. 2, 3, 19, 20).

[Use18] Usenko, Vladyslav; Demmel, Nikolaus and Cremers, Daniel: "The double sphere camera model". In: *2018 International Conference on 3D Vision (3DV).* IEEE. 2018, pp. 552–560 (cit. on pp. 28, 62, 66, 71).

[Val17] Valois, Jean-Sebastien; McAllister Bradley, David; Watson, Adam Charles; Melick, Peter Anthony and Miller, Andrew Gilbert: "Vehicle sensor calibration system". US 20170343654A1. 2017 (cit. on p. 4).

[Vas20] Vasiljevic, Igor; Guizilini, Vitor; Ambrus, Rares; Pillai, Sudeep; Burgard, Wolfram; Shakhnarovich, Greg and Gaidon, Adrien: "Neural ray surfaces for self-supervised learning of depth and ego-motion". In: *2020 International Conference on 3D Vision (3DV).* IEEE. 2020, pp. 1–11 (cit. on p. 76).

[Vis21] Vision, Allied: Manta G-235. https://www.alliedvision.com/en/products/cameras/detail/Manta/G-235.html. Retrieved on January 11th, 2021. 2021 (cit. on p. 33).

[WAK20] WAKAI, NOBUHIKO; AZUMA, TAKEO; NOBORI, KUNIO and SATO, SATOSHI: "CAMERA CALIBRATION METHOD, RECORDING MEDIUM, AND CAMERA CALIBRATION APPARATUS". US10664998 B2. 2020 (cit. on p. 4).

[Wan20] Wang, Wenshan; Zhu, Delong; Wang, Xiangwei; Hu, Yaoyu; Qiu, Yuheng; Wang, Chen; Hu, Yafei; Kapoor, Ashish and Scherer, Sebastian: "Tartanair: A dataset to push the limits of visual slam". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 4909–4916 (cit. on pp. 88, 89, 92).

[Wei20] Wei, Xingkui; Zhang, Yinda; Li, Zhuwen; Fu, Yanwei and Xue, Xiangyang: "Deepsfm: Structure from motion via deep bundle adjustment". In: *European conference on computer vision*. Springer. 2020, pp. 230–247 (cit. on p. 79).

[Xia17] Xiao, Suzhi; Tao, Wei and Zhao, Hui: "Flexible and accurate camera calibration using imperfect planar target". In: *International Conference on Optical and Photonics Engineering (icOPEN 2016)*. Vol. 10250. International Society for Optics and Photonics. 2017, p. 102501C (cit. on p. 63).

[Xue19] Xue, Zhucun; Xue, Nan; Xia, Gui-Song and Shen, Weiming: "Learning to calibrate straight lines for fisheye image rectification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1643–1651 (cit. on pp. 77, 78).

[Yak18] Yakura, Hiromu; Shinozaki, Shinnosuke; Nishimura, Reon; Oyama, Yoshihiro and Sakuma, Jun: "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism". In: *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 2018, pp. 127–134 (cit. on p. 128).

[Yan17] Yan, Han; Zhang, Yu; Zhang, Shunli; Zhao, Sicong and Zhang, Li: "Focal length estimation guided with object distribution on FocaLens dataset". In: *Journal of Electronic Imaging* 26.3 (2017), pp. 033018–033018 (cit. on p. 75).

[Yin18] Yin, Xiaoqing; Wang, Xinchao; Yu, Jun; Zhang, Maojun; Fua, Pascal and Tao, Dacheng: "Fisheyerecnet: A multi-context collaborative deep network for fisheye image rectification". In: *Proceedings of the European conference on computer vision (ECCV).* 2018, pp. 469–484 (cit. on p. 77).

[Zha00] Zhang, Zhengyou: "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334 (cit. on pp. 2, 3, 19, 20).

[Zhu19] Zhuang, Bingbing; Tran, Quoc-Huy; Lee, Gim Hee; Cheong, Loong Fah and Chandraker, Manmohan: "Degeneracy in self-calibration revisited and a deep learning solution for uncalibrated slam". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. 2019, pp. 3766–3773 (cit. on p. 78).

[Zuc01] Zucchelli, Marco and Kosecka, Jana: "Motion bias and structure distortion induced by calibration errors." In: *BMVC.* Citeseer. 2001, pp. 1–10 (cit. on pp. 4, 27).

A Additional fundamentals

This appendix addresses additional fundamentals that were excluded in Chap. 2 due to conciseness considerations. Sec. A.1 introduces the background of deep neural networks, which are applied in Chap. 6. Secs. A.2-A.3 provide additional details on least squares estimation and the Gauss-Newton algorithm. Finally, Sec. A.4 gives an overview of the representations of rotations in 3D space, and Sec. A.5 addresses the inversion of polynomial distortion models which is employed throughout this thesis.

A.1 Deep neural networks

Deep neural networks are a central technique in modern computer vision, complementing the classical computer vision techniques that rely on analytically derived geometric relations and handcrafted features. In general, deep neural networks solve computer vision tasks by learning from data [LeC15, Goo16]. As such, they have outperformed classical algorithms in many tasks, including object detection [Red16], semantic segmentation [Ron15], stereo depth estimation [Lag20], and optical flow estimation [Tee20]. While we refer to [Goo16] for an in-depth introduction to deep neural networks and deep learning, this section briefly introduces the basic terms related to deep neural networks that are used within this thesis.

We use the term *deep neural network* $\mathcal{N}$ to denote a mathematical model $\mathbf{f}(\mathbf{X}; \mathbf{w})$ that maps an input variable $\mathbf{X}$ to an output variable $\mathbf{Y}$ by passing the input through multiple transformations, also referred to as the network's *layers*. The model parameters $\mathbf{w}$ of a deep neural network are referred to

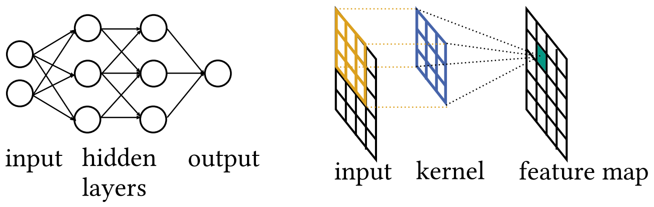


Figure A.1: Schematic of a feed-forward deep neural network (left, depiction inspired by [LeC15]) and a convolutional layer (right, depiction inspired by [Yak18]).

as its *weights*. The weights of a neural network are obtained through *training*, where the cost function of classical optimization problems (Sec. 2.2) is replaced by a *loss function* to be minimized. During training, the gradients of the loss w.r.t. all weights are computed using the *backpropagation algorithm*, and the weights are updated to minimize the loss. In the following, we introduce two specific types of deep neural networks, namely *convolutional neural networks (CNNs)* and *gated recurrent units (GRUs)*, as they are used in the course of this thesis.

A.1.1 Convolutional neural networks

Convolutional neural networks (CNNs) are among the most prevalent types of deep neural networks in computer vision. CNNs are generally designed for data that has a grid structure and their key building block are convolutional layers that perform a discretized version of the mathematical convolution operation (see Fig. A.1) [Goo16, p.326-333].

A convolutional layer for image processing is defined by a set of two-dimensional filters, where the weights $\mathbf{w}$ of the network determine the filter kernel. Depending on the kernel, a convolutional layer is capable of extracting edges, but also high-level shapes, such as faces, from an image. The output of a convolutional layer is referred to as the *feature maps*.

Besides convolutional layers, a CNN typically contains pooling layers that reduce the dimension of the feature maps, and non-linear activation functions, such as a ReLU or Sigmoid function [LeC15]. CNNs are oftentimes employed as the *backbone* for advanced large-scale computer vision models. This means that a CNN is used to extract features from the images which are subsequently processed by other parts of the overall model (see e.g. Fig. 6.5).

A.1.2 Gated recurrent units

Unlike feed-forward networks, which only pass information in one direction, *recurrent neural networks (RNNs)* contain feedback connections that feed information on prior outputs back into the model [LeC15]. They are commonly used for processing sequential data, such as language, in which information on prior sequence elements (e.g. prior words in a given sentence) is needed to correctly interpret the meaning of the current element (e.g. the current word) [LeC15].

Practically, feedback connections are implemented in terms of a *hidden state* $\mathbf{h}$. The hidden state is a vector that contains information on the history of previous sequence elements. The RNN's output for a given element will not only depend on the element itself but also on the hidden state. Thereby, the hidden state (i) provides context for processing the current element, and (ii) will, after processing the full sequence, contain information on the sequence as a whole [LeC15].

Two prevalent types of recurrent neural networks are *long short-term memory networks (LSTMs)* [Hoc97] and *gated recurrent units (GRUs)* [Cho14]. Both architectures have been developed to enable long-term memory, which had been a challenge for the original implementations of RNNs due to vanishing or exploding gradients [Chu14].

A GRU does not directly update the hidden state in every iteration t, but it contains an *update gate* that determines how much the new *candidate hidden state* $\tilde{\mathbf{h}}_t$ should contribute to the new hidden state $\mathbf{h}_t$ [Chu14]. The candidate hidden state $\tilde{\mathbf{h}}_t$ is computed as

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_\mathbf{h}\mathbf{x}_t + \mathbf{U}(\mathbf{c}_t \odot \mathbf{h}_{t-1})), \tag{A.1}$$

where $\mathbf{x}_t$ is the input, $\mathbf{W_h}$ and $\mathbf{U}$ contain the model weights and $\odot$ denotes the element-wise multiplication [Bal15]. The reset gate $\mathbf{c}_t$ determines how much $\tilde{\mathbf{h}}_t$ is influenced by the sequence history. Specifically, setting $\mathbf{c}_t = 0$, will remove any impact of the history. The reset gate is given by $\mathbf{c}_t = \tilde{\sigma}(\mathbf{W_c}\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1})$ with $\tilde{\sigma}(\cdot)$ denoting a logistic sigmoid function.

Finally, the new hidden state $\mathbf{h}_t$ is a weighted superposition of the previous hidden state $\mathbf{h}_{t-1}$ and the candidate hidden state $\tilde{\mathbf{h}}_t$

$$\mathbf{h}_t = (1 - \mathbf{s}_t) \odot \mathbf{h}_{t-1} + \mathbf{s}_t \odot \tilde{\mathbf{h}}_t, \tag{A.2}$$

where the update gate $\mathbf{s}_t$ is given by $\mathbf{s}_t = \tilde{\sigma}(\mathbf{W_s}\mathbf{x}_t + \mathbf{U_s}\mathbf{h}_{t-1})$ [Bal15].

A *convolutional gated recurrent unit (convGRU)* [Bal15] is a variant of a GRU that replaces all matrix multiplications in Eqs. (A.1-A.2) with a convolution operation $\otimes$. The update operation thus becomes [Bal15]:

$$\begin{aligned}
\mathbf{s}_t &= \tilde{\sigma}(\mathbf{W_s} \otimes \mathbf{x}_t + \mathbf{U_s} \otimes \mathbf{h}_{t-1}),\\
\mathbf{c}_t &= \tilde{\sigma}(\mathbf{W_c} \otimes \mathbf{x}_t + \mathbf{U}_c \otimes \mathbf{h}_{t-1}),\\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W_h} \otimes \mathbf{x}_t + \mathbf{U} \otimes (\mathbf{c}_t \odot \mathbf{h}_{t-1})),\\
\mathbf{h}_t &= (1 - \mathbf{s}_t) \odot \mathbf{h}_{t-1} + \mathbf{s}_t \odot \tilde{\mathbf{h}}_t.
\end{aligned} \tag{A.3}$$

While the formulas above define two basic versions of a GRU and a convGRU, different works may use different variants of these formulations. The convGRU used in Chap. 6 is a modification of Eq. (A.3), explained in App. Sec. C.1.

A.2 Linear least squares

In the following, the solution of linear least squares problems is shown and the Gauss-Newton update (2.9), which builds upon this solution, is derived.

As stated in Eq. (2.6), a least squares problem is determined by a cost function $E(\boldsymbol{\beta}) = \frac{1}{2}\mathbf{r}(\boldsymbol{\beta})^\mathsf{T}\mathbf{r}(\boldsymbol{\beta})$, where $\mathbf{r}(\boldsymbol{\beta}) \in \mathbb{R}^n$ is the vector of residuals. In model fitting, the vector of residuals can be written as $\mathbf{r}(\boldsymbol{\beta}) = \mathbf{f}(\mathbf{X}, \boldsymbol{\beta}) - \mathbf{Y}^*$, where $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta})$ is the model prediction and $\mathbf{Y}^*$ the observations (see Sec. 2.2).

In *linear least squares (LLS)* problems, the function $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta})$ is linear in the parameters $\boldsymbol{\beta}$, and can thus be written as $\mathbf{f}(\mathbf{X}, \boldsymbol{\beta}) = \mathbf{A}\boldsymbol{\beta}$ with $\mathbf{A} \in \mathbb{R}^{d \times n}$ [Noc06, p.250]. Then, the quadratic cost function can be written as

$$\text{LLS problem}: \quad E(\boldsymbol{\beta}) = \frac{1}{2}(\mathbf{A}\boldsymbol{\beta} - \mathbf{Y}^*)^\mathsf{T}(\mathbf{A}\boldsymbol{\beta} - \mathbf{Y}^*). \tag{A.4}$$

Linear least squares problems can be solved analytically, by setting $\nabla E(\boldsymbol{\beta}) \overset{!}{=} 0$ [Noc06, p.250]. The solution is then defined by the *normal equations*,

$$(\mathbf{A}^\mathsf{T}\mathbf{A})\hat{\boldsymbol{\beta}} = \mathbf{A}^\mathsf{T}\mathbf{Y}^*, \tag{A.5}$$

which can be solved for $\hat{\boldsymbol{\beta}}$ by computing the inverse $(\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}$ explicitly, or, more commonly, by using algorithms such as *QR*- or Cholesky-decomposition [Noc06, p.250-253]. If the measurement noise $\boldsymbol{\epsilon}$ is independently and identically distributed (i.i.d.), following a Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\epsilon\epsilon})$ with covariance matrix $\boldsymbol{\Sigma}_{\epsilon\epsilon} = \mathrm{diag}(\sigma^2) \in \mathbb{R}^{n \times n}$ and variance $\sigma^2 \in \mathbb{R}$, the least squares solution is also the maximum likelihood estimator [Noc06, p.250].

A.3 Derivation of the Gauss-Newton algorithm

The idea behind the Gauss-Newton algorithm (2.9) for solving a non-linear least squares problem (2.8) is to locally approximate the residuals with a linear function which gives local solutions via (A.5) [Noc06, p.255]. The local solutions serve as iterative update steps towards convergence. To obtain the Gauss-Newton step, the Taylor expansion of the residuals can be written up to first order as [Noc06, p.255]:

$$\mathbf{r}(\boldsymbol{\beta}_0 + \boldsymbol{\Delta\beta}) \approx \mathbf{r}(\boldsymbol{\beta}_0) + \mathbf{J}\boldsymbol{\Delta\beta}, \tag{A.6}$$

where $\mathbf{J} \in \mathbb{R}^{n \times d}$ denotes the Jacobian (2.10) of the residuals w.r.t. the parameters, evaluated at $\boldsymbol{\beta}_0$. Using this linearization, the cost function $E(\cdot)$ of a non-linear least squares problem (2.8) can be approximated around $\boldsymbol{\beta} = \boldsymbol{\beta}_0$:

$$E(\boldsymbol{\beta}_0 + \boldsymbol{\Delta\beta}) \approx \tilde{E}(\boldsymbol{\beta}_0 + \boldsymbol{\Delta\beta}) = \frac{1}{2}(\mathbf{J}\boldsymbol{\Delta\beta} + \mathbf{r}(\boldsymbol{\beta}_0))^\top \mathbf{W}(\mathbf{J}\boldsymbol{\Delta\beta} + \mathbf{r}(\boldsymbol{\beta}_0)). \quad \text{(A.7)}$$

The parameter update $\boldsymbol{\Delta\beta}$ is computed so as to minimize the approximated cost function. This is achieved by setting its gradient to zero:

$$\nabla\tilde{E}(\boldsymbol{\beta}_0 + \boldsymbol{\Delta\beta}) = \mathbf{J}^\top \mathbf{W}(\mathbf{J}\boldsymbol{\Delta\beta} + \mathbf{r}(\boldsymbol{\beta}_0)) \overset{!}{=} 0. \quad \text{(A.8)}$$

As this is performed in each iteration k, we replace $\boldsymbol{\beta}_0$ with $\boldsymbol{\beta}^k$ and $\mathbf{r}(\boldsymbol{\beta}_0)$ with $\mathbf{r}^k$. Then, rearranging Eq. (A.8) gives the *Gauss-Newton update* [Noc06, p.254]

$$\begin{aligned} \mathbf{J}^\top \mathbf{W}\mathbf{J}\boldsymbol{\Delta\beta} &= -\mathbf{J}^\top \mathbf{W}\mathbf{r}^k \\ \boldsymbol{\beta}^{k+1} &= \boldsymbol{\beta}^k + \boldsymbol{\Delta\beta}. \end{aligned} \quad \text{(A.9)}$$

The Gauss-Newton update is performed iteratively until a pre-defined convergence threshold is reached.

A.4 Transformations in 3D

Representing the geometry of the 3D world relative to different coordinate systems requires pose transformations. In general, a pose transformation in three dimensions $\mathbf{G} \in SE(3)$ can be described by a translation, represented by a vector $\mathbf{t} \in \mathbb{R}^3$, and a rotation which can be represented in different ways, including [För16, p.325-326][Bec21]:

1 **Rotation matrix $\mathbf{R} \in SO(3)$.** In the rotation matrix representation, a 3D rotation is expressed in terms of a 3×3 matrix $\mathbf{R}$ with $\mathbf{R}^\top = \mathbf{R}^{-1}$ and $\det(\mathbf{R}) = 1$. The rotated coordinates $\mathbf{x}'$ of a 3D point $\mathbf{x}$ are obtained by multiplication $\mathbf{x}' = \mathbf{R}\mathbf{x}$. The rotation matrix representation is less suited for optimization, as it is overparametrized and requires enforcing the abovementioned properties during optimization [Bec21].

2 **Euler angles.** In the Euler angle representation, a rotation is expressed in terms of three angles ϕ_1, ϕ_2, ϕ_3, one around each coordinate axis. The rotated coordinates $\mathbf{x}'$ of a point $\mathbf{x}$ can be obtained by converting the angles into a rotation matrix, e.g. $\mathbf{R} = \mathbf{R}(\phi_3)\mathbf{R}(\phi_2)\mathbf{R}(\phi_1)$. While Euler angles are an intuitive representation, they suffer from Gimbal lock which limits their practical applicability [Bec21].

3 **Axis-angle representation.** In the axis-angle representation, a rotation is expressed in terms of an axis $\mathbf{v} \in \mathbb{R}^3$ and a rotation angle $\alpha \in \mathbb{R}$ around the axis. The axis-angle representation consists of a single vector $\mathbf{v} \in \mathbb{R}^3$, where the direction $\tilde{\mathbf{v}} = \mathbf{v}/||\mathbf{v}||$ defines the rotation axis and the norm $\alpha = ||\mathbf{v}||$ quantifies the angle[1]. The rotated coordinates $\mathbf{x}'$ of a point $\mathbf{x}$ are obtained via [För16, p.331]

$$\mathbf{x}' = \mathbf{x}\cos(\alpha) + (\tilde{\mathbf{v}} \times \mathbf{x})\sin(\alpha) + \tilde{\mathbf{v}}(\tilde{\mathbf{v}} \cdot \mathbf{x})(1 - \cos(\alpha)). \tag{A.10}$$

The axis-angle representation is suitable for optimization, as it avoids the overparametrization of other representations. However, it comes with the disadvantage of requiring trigonometric functions, and the case of zero rotation must be explicitly handled.

4 **Quaternion representation.** In the quaternion representation, a rotation is uniquely defined by a four-component vector $\mathbf{q} = (q_1, q_2, q_3, q_4)^\top \in \mathbb{H}$ with $||\mathbf{q}|| = 1$, i.e. a unit quaternion. The quaternion representation avoids singularities which is beneficial for optimization [För16, p.333]. As the four-component vector is overparametrized, a three-component local parametrization (see e.g. [Bla21]) is often used during optimization.

A.5 Inversion of polynomial distortion models

While the pinhole model and the unified camera model can be inverted analytically (see Sec. 2.1), polynomial distortion models $d(\rho) = \rho(1 + \kappa_1\rho^2 +$

[1] The case of zero rotation must be explicitly implemented, as $||\mathbf{v}|| = 0$ would otherwise yield a division by zero.

$\kappa_2\rho^4 + \kappa_3\rho^6)$ must be inverted numerically each given image point $\mathbf{u}$. One way to achieve this is by first applying the inverse pinhole projection $\tilde{\mathbf{x}} = \left(\frac{u-c_x}{f_x}, \frac{v-c_y}{f_y}\right)^\top$, giving the distorted distance $\tilde{\rho} = \sqrt{\tilde{x}^2 + \tilde{y}^2}$. Then the undistorted distance ρ can be found by iteratively performing Newton steps [Noc06, p.44] to minimize $\rho = \arg\min_{\rho' \in \mathbb{R}^+} d(\rho') - \tilde{\rho}$.

A.6 Structure-from-Motion and visual SLAM

Structure-from-Motion (SfM) and visual SLAM are techniques to infer 3D structure and camera poses from a set of images. As an in-depth introduction to Structure-from-Motion and visual SLAM is beyond the scope of this thesis, this section only gives a brief overview and refers to [Sch16a, Cam21] for detailed descriptions of the techniques.

A Structure-from-Motion pipeline, such as COLMAP [Sch16a], can be divided into two main steps:

1 **Correspondence search** First, the coordinates of outstanding points in the individual images are extracted (see e.g. tip of the roof in Fig 2.5). This is called keypoint or *feature extraction.* Each keypoint is assigned a feature descriptor that characterizes the image patch around the point. During *feature matching*, the descriptors are used to find corresponding keypoints across images.

2 **Reconstruction** Given the corresponding points across images, camera poses and 3D structure are reconstructed. Initial estimates for all parameters are typically obtained through epipolar geometry and triangulation (see e.g. [Sch16a]), followed by a parameter refinement during bundle adjustment (Sec. 2.4.2). In addition, modern SfM pipelines incorporate advanced outlier filtering techniques to reduce the impact of false matches (see e.g. [Sch16a]).

Visual SLAM is closely related to Structure-from-Motion, but originates in the field of robotics [Tak17, Mur15, Cam21, Tee21]. SLAM, simultaneous localization and mapping, generally refers to systems that use a sequence of measurements of one or multiple sensors to estimate the sensor's or robot's current location and to construct a map of the environment. *Visual* SLAM performs this task using one or multiple cameras. It can be classified as a special case of the SfM problem, in which the input images are an ordered sequence taken by the same camera(s) at different points in time, typically aiming at real-time applications. Visual SLAM can be realized with different frameworks as well as different optimization techniques, and we refer to reviews [Tak17] and [Mac22] for a comprehensive overview.

B Derivations

B.1 The expected mapping error

This section provides a more detailed derivation of the expected mapping error (EME) introduced in Sec. 4.3, as adapted from [Hag22b, Hag20c]. The EME is defined by the expected value of a quadratic mapping error $K(\hat{\theta}, \bar{\theta})$, where $K(\hat{\theta}, \bar{\theta})$ describes the deviation between the projection based on the estimated intrinsics $\pi_C(\mathbf{x}, \hat{\theta})$ and the projection based on the ground-truth intrinsics $\pi_C(\mathbf{x}, \bar{\theta})$. The derivation does not depend on the exact formulation of $K(\hat{\theta}, \bar{\theta})$, as long as it can be approximated with a Taylor expansion up to second order.

B.1.1 Approximation of the mapping error

The derivation of the EME relies on an approximation of $K(\hat{\theta}, \bar{\theta})$, using a Taylor expansion around the true intrinsics $\hat{\theta} = \bar{\theta}$ [Hag22b, Hag20c]:

$$\begin{aligned} K(\hat{\theta}, \bar{\theta}) &= \frac{1}{2n_{\mathcal{G}}} \mathbf{r}_K(\hat{\theta}, \bar{\theta})^\top \mathbf{r}_K(\hat{\theta}, \bar{\theta}) \\ &\approx K(\bar{\theta}, \bar{\theta}) + \nabla K \boldsymbol{\Delta\theta} + \frac{1}{2} \boldsymbol{\Delta\theta}^\top \mathbf{H}_K \boldsymbol{\Delta\theta}, \end{aligned} \tag{B.1}$$

where, $\mathbf{r}_K(\hat{\theta}, \bar{\theta})$ are the mapping residuals, $\boldsymbol{\Delta\theta} = \hat{\theta} - \bar{\theta}$ is the parameter deviation and $2n_{\mathcal{G}}$ is the number of residuals. The gradient and the Hessian of

K w.r.t. $\boldsymbol{\Delta\theta}$ are given by

$$\begin{aligned} \nabla K &= \frac{dK}{d\boldsymbol{\Delta\theta}} = 2\frac{1}{2n_g}\mathbf{r}_K(\hat{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}})^\top \mathbf{J}_{\text{res}} \\ \mathbf{H}_K &= \frac{d^2K}{d\boldsymbol{\Delta\theta}^2} = 2\frac{1}{2n_g}{\mathbf{J}_{\text{res}}}^\top \mathbf{J}_{\text{res}} + \text{higher order terms,} \end{aligned} \tag{B.2}$$

where $\mathbf{J}_{\text{res}} = d\mathbf{r}_K/d\boldsymbol{\Delta\theta}$ denotes the Jacobian of the mapping residuals.

Using the Gauss-Newton approximation $d^2\mathbf{r}_K/d\boldsymbol{\Delta\theta}^2 \approx 0$, the higher-order terms of the Hessian vanish [Tri00, p. 16]. The mapping error $K(\hat{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}})$ can then be approximated as follows [Hag22b, Hag20c]:

$$\begin{aligned} K(\hat{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}}) &\approx K(\bar{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}}) + \frac{2}{2n_g}\mathbf{r}_K(\bar{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}})^\top \mathbf{J}_{\text{res}}\boldsymbol{\Delta\theta} + \frac{1}{4n_g}\boldsymbol{\Delta\theta}^\top(2{\mathbf{J}_{\text{res}}}^\top \mathbf{J}_{\text{res}})\boldsymbol{\Delta\theta} \\ &\approx \frac{1}{2n_g}\boldsymbol{\Delta\theta}^\top({\mathbf{J}_{\text{res}}}^\top \mathbf{J}_{\text{res}})\boldsymbol{\Delta\theta} \\ &\approx \boldsymbol{\Delta\theta}^\top \mathbf{H}\boldsymbol{\Delta\theta}, \end{aligned} \tag{B.3}$$

where $\mathbf{H} := \frac{1}{2n_g}{\mathbf{J}_{\text{res}}}^\top \mathbf{J}_{\text{res}}$ is introduced, denoted as the *model matrix*. Here the second step uses that $\mathbf{r}_K(\bar{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}}) = \mathbf{0}$ and $K(\bar{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}}) = 0$.

B.1.2 Derivation of the distribution of the mapping error

The deviation $\boldsymbol{\Delta\theta} = \hat{\boldsymbol{\theta}} - \bar{\boldsymbol{\theta}}$ of estimated intrinsics from the true intrinsics is a random variable that depends on the data sample used for calibration. As a consequence, $K(\hat{\boldsymbol{\theta}},\bar{\boldsymbol{\theta}})$ is also a random variable and its distribution can be derived based on the distribution of $\boldsymbol{\Delta\theta}$.

As calibration relies on least squares estimation, estimated model parameters $\hat{\boldsymbol{\theta}}$ asymptotically follow a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}$ and covariance $\boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}\hat{\boldsymbol{\theta}}}$ [Tri00, p.7-9]. Consequently, the parameter error $\boldsymbol{\Delta\theta}$ follows a multivariate Gaussian distribution as well, with mean $\boldsymbol{\mu}_{\boldsymbol{\Delta\theta}} = \mathbf{0}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{\Delta\theta}\boldsymbol{\Delta\theta}} = \boldsymbol{\Sigma}_{\hat{\boldsymbol{\theta}}\hat{\boldsymbol{\theta}}}$.

To derive the distribution of $K(\hat{\theta}, \bar{\theta})$, one can re-write $\boldsymbol{\Delta\theta}$ in terms of a multivariate standard Gaussian distributed random variable $\mathbf{a}$:

$$\begin{aligned} &\mathbf{a} := \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{-1/2}(\boldsymbol{\Delta\theta} - \boldsymbol{\mu}_{\Delta\theta}) \\ \Leftrightarrow \quad &\boldsymbol{\Delta\theta} = \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{1/2}\mathbf{a} + \boldsymbol{\mu}_{\Delta\theta} \end{aligned} \tag{B.4}$$

Using $\boldsymbol{\mu}_{\Delta\theta} = \mathbf{0}$ and substituting this expression in (B.3) yields

$$\begin{aligned} K &= \boldsymbol{\Delta\theta}^\top \mathbf{H} \boldsymbol{\Delta\theta} \\ &= \mathbf{a}^\top \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} \mathbf{H} \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} \mathbf{a}. \end{aligned} \tag{B.5}$$

The matrix $\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} \mathbf{H} \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}}$ can now be diagonalized using the spectral theorem. Specifically, one can rewrite $\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} \mathbf{H} \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} = \mathbf{P}^\top \boldsymbol{\Lambda} \mathbf{P}$ where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} \mathbf{H} \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}}$ and $\mathbf{P}$ is an orthogonal matrix ($\mathbf{P}^\top \mathbf{P} = \mathbf{I}$) where the columns are the corresponding eigenvectors. Substitution in (B.5) then yields

$$\begin{aligned} K &= \mathbf{a}^\top \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} \mathbf{H} \boldsymbol{\Sigma}_{\hat{\theta}\hat{\theta}}{}^{\frac{1}{2}} \mathbf{a} \\ &= \mathbf{a}^\top \mathbf{P}^\top \boldsymbol{\Lambda} \mathbf{P} \mathbf{a} \\ &= \mathbf{c}^\top \boldsymbol{\Lambda} \mathbf{c}. \end{aligned} \tag{B.6}$$

Here the last step is obtained by defining $\mathbf{c} := \mathbf{Pa}$. Since $\mathbf{P}$ is orthogonal and $\mathbf{a}$ is a multivariate standard Gaussian distributed random variable, it follows that $\mathbf{c}$ is a multivariate standard Gaussian distributed random variable as well. Then, since $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues $\lambda_{n=1,\ldots,N_\theta}$ of the matrix $\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}\mathbf{H}\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}$, expression (B.6) can be re-written in terms of a sum:

$$\begin{aligned} K &= \mathbf{c}^\top \mathbf{\Lambda} \mathbf{c} \\ &= \sum_{i=n}^{N_\theta} \lambda_n c_n^2 \quad \text{with} \quad c_n \sim \mathcal{N}(0,1) \\ &= \sum_{n=1}^{N_\theta} \lambda_n Q_n, \quad \text{with} \quad Q_n \sim \chi^2(1), \end{aligned} \tag{B.7}$$

where N_θ is the number of eigenvalues λ_n, corresponding to the number of parameters in $\boldsymbol{\theta}$. Expression B.7 is a linear combination of central first-order χ^2-distributed random variables, weighted by the eigenvalues λ_n.

Based on this expression, the expectation value of K can directly be derived. Given that the expectation value of a χ^2-distribution with one degree of freedom is given by $\mathbb{E}[\chi^2(1)] = 1$, it follows that [Hag22b, Hag20c]:

$$\begin{aligned} \mathbb{E}[K(\hat{\boldsymbol{\theta}}, \bar{\boldsymbol{\theta}})] &= \mathbb{E}[\sum_{n=1}^{N_\theta} \lambda_n Q_n] = \sum_{n=1}^{N_\theta} \lambda_n \mathbb{E}[Q_n] = \sum_{n=1}^{N_\theta} \lambda_n \\ &= \operatorname{trace}(\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}\mathbf{H}\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}^{\frac{1}{2}}) \\ &= \operatorname{trace}(\mathbf{\Sigma}_{\hat{\theta}\hat{\theta}}\mathbf{H}). \end{aligned}$$

This is the proposed uncertainty metric, referred to as the expected mapping error EME [Hag22b, Hag20c].

B.2 Intrinsics Jacobian for self-calibrating bundle adjustment

This section derives the intrinsics Jacobian required for the self-calibrating bundle adjustment (SC-BA) layer introduced in Chap. 6. This derivation is adapted from [Hag23].

The SC-BA layer optimizes camera poses, depth, and intrinsics by minimizing the reprojection error. Let $\mathbf{u}_i$ be a point in image $\mathbf{I}_i$ and $\mathbf{u}^*_{ij}$ the measured corresponding point in image $\mathbf{I}_j$. Then the associated optimization residual is given by

$$\begin{aligned} \mathbf{r}_{ij} &= \mathbf{u}^*_{ij} - \pi_C(\mathbf{G}_{ij} * \pi_C^{-1}(\mathbf{u}_i, z_i, \theta), \theta) && \text{(B.8)} \\ &= \mathbf{u}^*_{ij} - \pi_C(\mathbf{x}_{ij}(\theta), \theta), && \text{(B.9)} \end{aligned}$$

where θ denotes the intrinsics, z_i denotes the depth and $\mathbf{G}_{ij} \in SE(3)$ denotes the relative 3D pose (cf. Sec. 2.3.1). The function π_C describes the projection from 3D to the image and π_C^{-1} the inverse projection (cf. Sec. 2.1). In the second step, we defined $\mathbf{x}_{ij}(\theta) := \mathbf{G}_{ij} * \pi_C^{-1}(\mathbf{u}_i, z_i, \theta)$ for better readability.

As (B.9) contains direct and indirect dependencies of θ, the Jacobian can be obtained using the total derivative [Hag23]:

$$\begin{aligned} \frac{d\mathbf{r}_{ij}}{d\theta} &= -\frac{d}{d\theta}\pi_C(\mathbf{x}_{ij}(\theta), \theta) && \text{(B.10)} \\ &= -\frac{\partial \pi_C(\mathbf{x}_{ij}, \theta)}{\partial \mathbf{x}_{ij}} \frac{d\mathbf{x}_{ij}(\theta)}{d\theta} - \frac{\partial \pi_C(\mathbf{x}_{ij}, \theta)}{\partial \theta} && \text{(B.11)} \end{aligned}$$

The derivative $\frac{d\mathbf{x}_{ij}(\theta)}{d\theta}$ is obtained by writing out the pose transformation in terms of the relative rotation $\mathbf{R}_{ij} \in SO(3)$ and translation $\mathbf{t}_{ij} \in \mathbb{R}^3$:

$$\begin{aligned} \mathbf{x}_{ij}(\theta) &= \mathbf{G}_{ij} * \pi_C^{-1}(\mathbf{u}_i, z_i, \theta) && \text{(B.12)} \\ &= \mathbf{R}_{ij}\pi_C^{-1}(\mathbf{u}_i, z_i, \theta) + \mathbf{t}_{ij}. && \text{(B.13)} \end{aligned}$$

The derivative is then given by

$$\frac{d\mathbf{x}_{ij}(\theta)}{d\theta} = \mathbf{R}_{ij} \frac{d\pi_C^{-1}(\mathbf{u}_i, z_i, \theta)}{d\theta}. \qquad \text{(B.14)}$$

Substituting this expression in (B.10) gives [Hag23]:

$$\frac{d\mathbf{r}_{ij}}{d\theta} = -\underbrace{\frac{\partial \pi_C(\mathbf{x}_{ij}, \theta)}{\partial \mathbf{x}_{ij}}}_{(A)} \mathbf{R}_{ij} \underbrace{\frac{d\pi_C^{-1}(\mathbf{u}_i, z_i, \theta)}{d\theta}}_{(B)} - \underbrace{\frac{\partial \pi_C(\mathbf{x}_{ij}, \theta)}{\partial \theta}}_{(C)}, \tag{B.15}$$

which holds independent of the specific projection model π_C (cf. Sec. 2.1).

To obtain the intrinsics Jacobian for the pinhole model, the individual terms (A),(B),(C) must be computed accordingly. For the pinhole model (cf. Sec. 2.1), projection π_C and inverse projection π_C^{-1} are given by

$$\pi_C(\mathbf{x}, \theta) = \begin{bmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{bmatrix} \quad \text{and} \quad \pi_C^{-1}(\mathbf{u}, \theta, z) = z \begin{bmatrix} \frac{u - c_x}{f_x} \\ \frac{v - c_y}{f_y} \\ 1 \end{bmatrix}, \tag{B.16}$$

where $\mathbf{x} = (x,y,z)^\top$ is a 3D point, $\mathbf{u} = (u, v)^\top$ is an image point, (c_x, c_y) denotes the principal point and f_x, f_y are the focal lengths. Computation of the derivatives yields the following matrices which can be substituted in Eq. (B.15) to obtain the overall Jacobian w.r.t. pinhole intrinsics [Hag23]:

$$\frac{\partial \pi_C(\mathbf{x}_{ij}, \theta)}{\partial \mathbf{x}_{ij}} = \begin{bmatrix} \frac{f_x}{z_{ij}} & 0 & -f_x \frac{x_{ij}}{z_{ij}^2} \\ 0 & \frac{f_y}{z_{ij}} & -f_y \frac{y_{ij}}{z_{ij}^2} \end{bmatrix} \tag{B.17}$$

$$\frac{d\pi_C^{-1}(\mathbf{u}_i, z_i, \theta)}{d\theta} = \begin{bmatrix} -z_i \frac{u - c_x}{f_x^2} & 0 & -\frac{z_i}{f_x} & 0 \\ 0 & -z_i \frac{v - c_y}{f_y^2} & 0 & -\frac{z_i}{f_y} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.18}$$

$$\frac{\partial \pi_C(\mathbf{x}_{ij}, \theta)}{\partial \theta} = \begin{bmatrix} \frac{x_{ij}}{z_{ij}} & 0 & 1 & 0 \\ 0 & \frac{y_{ij}}{z_{ij}} & 0 & 1 \end{bmatrix}. \tag{B.19}$$

The Jacobian for the unified camera model (2.4) can be derived analogously, however, the individual derivatives contain significantly longer expressions. In our implementation of the unified camera model, which we only use during inference, the Jacobian w.r.t. the unified model intrinsics is therefore approximated numerically.

C Methodological details

This appendix provides additional methodological details which were omitted in the main thesis due to conciseness considerations. This includes details on the model architecture of the self-calibration approach, as well as details on the baseline evaluation in Chap. 6.

C.1 DROID-SLAM model architecture and SLAM system

C.1.1 Neural network architectures

The model architecture used in Sec. 6.3 (see Fig. 6.5) consists of different modules, which were proposed in [Tee21] and adapted from [Tee20]. Specifically, it contains CNN modules used for feature and context extraction, and a convGRU. In the following, the underlying architectures and operations are described in further detail.

The architecture of the feature extraction and the context module is depicted in Fig. C.1 (A). It consists of convolutional layers and residual blocks (i.e. the building blocks of the ResNet architecture [He16]). The feature extraction module yields feature maps $\mathcal{F}_{\text{feat},i} \in \mathbb{R}^{128\times\tilde{H}\times\tilde{W}}$ for every image $\mathbf{I}_i$, where $\tilde{H} = H/8$ and $\tilde{W} = W/8$. The context module produces feature maps $\mathcal{F}_{\text{ini+con},i} \in \mathbb{R}^{256\times\tilde{H}\times\tilde{W}}$, where the first 128 channels provide the initialization for the hidden state, and the other 128 channels are the context features $\mathcal{F}_{\text{con}} \in \mathbb{R}^{N_{\mathcal{P}}\times 128\times\tilde{H}\times\tilde{W}}$.

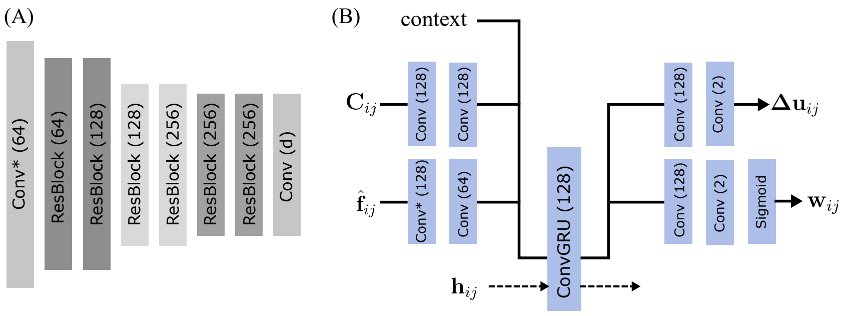


Figure C.1: Network architectures of DROID-SLAM [Tee21] modules that are used in the intrinsic self-calibration approach. Depiction adapted from [Tee21]. (A) CNN for feature extraction and context encoding. *Conv (d)* denotes a convolutional layer with d channels. All convolutional layers use 3×3 filter kernels, except those denoted by *, which use 7×7 filter kernels. In the feature extraction module, the final number of channels is $d = 128$, and in the context module $d = 256$. (B) Convolutional layers and ConvGRU in the update operator.

The convGRU is depicted in Fig. C.1 (B). Correlation $\mathbf{C}_{ij}$ and flow $\hat{\mathbf{f}}_{ij}$ are first passed through a set of convolutional layers, while the context features are directly injected into the convGRU. The input $\mathbf{x}_t$ to the convGRU is thus a stacked tensor of flow features $\mathcal{F}_{\text{flow}} \in \mathbb{R}^{N_{\mathcal{P}} \times 64 \times \tilde{H} \times \tilde{W}}$, correlation features $\mathcal{F}_{\text{corr}} \in \mathbb{R}^{N_{\mathcal{P}} \times 128 \times \tilde{H} \times \tilde{W}}$, and context features $\mathcal{F}_{\text{con}} \in \mathbb{R}^{N_{\mathcal{P}} \times 128 \times \tilde{H} \times \tilde{W}}$ where $N_{\mathcal{P}}$ is the number of image pairs in the current graph.

The hidden state[1] $\mathbf{h}_t \in \mathbb{R}^{N_{\mathcal{P}} \times 128 \times \tilde{H} \times \tilde{W}}$ is initialized with the first 128 channels from the context features $\mathcal{F}_{\text{ini+con},i}$ of each pair's first image $\mathbf{I}_i$ and updated in every iteration t through the following operations [Tee21, Tee20]:

$$\begin{aligned}
\mathbf{s}_t &= \tilde{\sigma}(\mathbf{W}_\mathbf{s} \otimes [\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{U}_\mathbf{s} \otimes \mathbf{g}_t),\\
\mathbf{c}_t &= \tilde{\sigma}(\mathbf{W}_\mathbf{c} \otimes [\mathbf{x}_t; \mathbf{h}_{t-1}] + \mathbf{U}_c \otimes \mathbf{g}_t),\\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_\mathbf{h} \otimes [\mathbf{x}_t; \mathbf{c}_t \odot \mathbf{h}_{t-1}] + \mathbf{U} \otimes \mathbf{g}_t),\\
\mathbf{h}_t &= (1 - \mathbf{s}_t) \odot \mathbf{h}_{t-1} + \mathbf{s}_t \odot \tilde{\mathbf{h}}_t,
\end{aligned} \tag{C.1}$$

[1] Following the convention introduced in Sec. A.1, we here use the index t to represent an update step; this index is equivalent to the index k used in Chap. 6.

which is a variant of the convGRU introduced in Sec. A.1. Here $\tilde{\sigma}(\cdot)$ is a Sigmoid function, $\odot$ denotes element-wise multiplication, and $\otimes$ denotes a convolution operation, where $\mathbf{W_s}, \mathbf{W_c}, \mathbf{W_h}, \mathbf{U}_s, \mathbf{U}_c, \mathbf{U}$ denote the associated learned parameters of the respective convolutional layer (see also source code provided along with [Tee21, Hag23]). The global state $\mathbf{g}_t \in \mathbb{R}^{N_{\mathcal{P}} \times 128}$ is computed by passing the hidden state through a convolutional layer with learned parameters $\mathbf{U_g}$ and subsequently averaging it across the image dimension [Tee21]:

$$\mathbf{g}'_t = \tilde{\sigma}(\mathbf{U_g} \otimes \mathbf{h}_{t-1}) \odot \mathbf{h}_{t-1} \in \mathbb{R}^{N_{\mathcal{P}} \times 128 \times \tilde{H} \times \tilde{W}} \tag{C.2}$$

$$\mathbf{g}_t = \frac{1}{\tilde{H} \times \tilde{W}} \sum_{\nu_W=1}^{\tilde{W}} \sum_{\nu_H=1}^{\tilde{H}} \mathbf{g}_t'^{\nu_H, \nu_W} \in \mathbb{R}^{N_{\mathcal{P}} \times 128}. \tag{C.3}$$

Thereby, it should provide global information across the image dimension [Tee21].

After updating the hidden state via (C.1), flow revisions $\mathbf{\Delta u}_{ij}$ and confidence weights $\mathbf{w}_{ij}$ are obtained by mapping the updated hidden state through two additional sets of convolutional layers (Fig. C.1 (B)). In addition, the damping factors for the depth block in the Levenberg Marquardt optimization (6.10), $\boldsymbol{\lambda}_\mathbf{z} \in \mathbb{R}_+^{\tilde{H} \times \tilde{W} \times N_{\mathcal{I}}}$, are obtained by averaging the hidden state of the convGRU across all image pairs $(\mathbf{I}_i, \mathbf{I}_j)$ with same source image $\mathbf{I}_i$, and passing the result through another convolutional and softplus layer [Tee21].

C.1.2 SLAM system

We employ the SLAM system from the DROID-SLAM [Tee21] open source implementation provided by the authors[1]. All hyperparameters of the SLAM system are taken from the original implementation and not specifically tuned for self-calibration. Likewise, the initialization of the system and the construction of the frame graph, which determines which image pairs are included

[1] https://github.com/princeton-vl/DROID-SLAM, commit 8016d2b9b72

during bundle adjustment, remains untouched. We therefore refer to [Tee21] for the underlying details.

C.2 Self-calibration baseline evaluation

This section provides additional details on the baseline evaluation in Chap. 6, as similarly reported in the supplementary of [Hag23].

C.2.1 Target-based calibration

To compare the self-calibration results with target-based calibration (Sec. 6.4.2), we used the calibration datasets provided along with the EuRoC [Bur16] and the TUM [Stu12b] datasets. Both calibration datasets consist of more than 500 images of a planar checkerboard target from which we randomly sampled calibration datasets of different sizes (20, 50, and 100 images), giving 20 datasets each. Before calibration, we undistorted all images using the reference intrinsics. Finally, target-based calibration is performed as described in Sec. 2.3.

C.2.2 COLMAP-based approaches

Evaluations of COLMAP were performed using hloc version 1.1 [Sar19] and COLMAP version 3.8 [Sch16a]. To select images for matching, we used the trained NetVLAD [Ara16] model provided by hloc [Sar19]. For feature extraction, we either used SIFT [Low04] or Superpoint [DeT18], using the pre-trained InLoc [Tai18] configuration provided in hloc. For feature matching, we either used nearest neighbour matching, using the default *nearest neighbor ratio* configuration from hloc [Sar19], or we used the pre-trained Superglue [Sar20] model provided in hloc. Before processing, all images were resized to the same dimensions as for DroidCalib (see Sec. 6.4).

As the performance of classical Structure-from-Motion pipelines depends on the selection of image pairs, we evaluated different strategies in terms of calibration accuracy and runtime (Tab. C.1). Our results suggest that exhaustive matching yields most accurate results, yet, it comes at the cost of high computation times that are practically unfeasible for long sequences (up to multiple days per sequence). Sequential matching with a window of five images yields the least accurate results but the shortest computation times. NetVLAD represents a compromise between accuracy and runtime (Tab. C.1).

Table C.1: Evaluation of COLMAP-based self-calibration on the TUM dataset, using different configurations. Values are Median [Min, Max] over all converged sequences of the respective configuration. Sequences associated with failed runs are only excluded from the respective failed configuration; not from all configurations. In Tab. 6.4, COLMAP+NetVLAD and COLMAP+NetVLAD+Superpoint+Superglue are reported, as a compromise between accuracy and runtime. Table extracted from [Hag23].

Method	ME (pixel)		Runtime		Failures
sequential	9.69	[2.42, 24.95]	2 min	[1 min, 6 min]	4 / 9
exhaustive	4.26	[2.46, 6.89]	2 h 19 min	[46 min, 2 days 18 h 15 min]	0 / 9
NetVLAD	6.54	[2.52, 52.1]	4 min	[2 min, 36 min]	0 / 9
NetVLAD+SP+SG	4.10	[1.66, 8.09]	13 min	[5 min, 51 min]	2 / 9

C.2.3 SelfSup-Calib

To evaluate the self-supervised learning approach to self-calibration [Fan22], we used the implementation provided by the authors[1]. We selected the model configuration for the unified camera model and kept the proposed resizing of images to an input size of 256×384. We also left all other hyperparameters unchanged. For the TUM and the TartanAir sequences, which were not evaluated in [Fan22], we used the same hyperparameters as for EuRoC.

To obtain results that are comparable across the different self-calibration methods, we trained the model separately for each sequence. This differs from the configuration used in [Fan22], which trained on a set of multiple sequences from the EuRoC dataset. Furthermore, the parametrization of

[1] https://github.com/TRI-ML/vidar.git, commit a306e23cc47ae05a0c35b0bf8acf7112c87c049c

the unified camera model in [Fan22] slightly differs from (2.4), however, the formulations are mathematically equivalent so that the model's final result can be directly converted into formulation (2.4) for evaluation.

As the calibration accuracy obtained with the described settings was significantly lower than the other self-calibration methods (Tab. 6.4), we further analyzed whether a larger number of training epochs or more training data could improve the results (Tab. C.2). Consistent with the results published in [Fan22], we find that training jointly on all EuRoC sequences yields significantly more accurate results than per-sequence training, but also significantly longer training times. Training per sequence, but increasing the number of epochs from 50 to 350 leads to slightly higher accuracy but also comes at the cost of a higher training time. Overall, this suggests that the self-supervised learning approach is better suited in settings where larger datasets are available, and that it suffers from the limited amount of data in the single-sequence setting.

Table C.2: Results of SelfSup-Calib [Fan22] using different settings. The reported values are Median [Min, Max] across all sequences in the respective dataset. For the *full dataset* setting we report the single result obtained by training 50 epochs jointly on all sequences. Table extracted from [Hag23].

Dataset	Method	ME (pixel)		Runtime
TartanAir	SelfSup-Calib (50 epochs)	18.3	[5.0, 60.1]	28 min
	SelfSup-Calib (350 epochs)	14.9	[2.95, 64.8]	3 h 08 min
	SelfSup-Calib (full dataset)	3.52	[–, –]	5 h 36 min
EuRoC	SelfSup-Calib (50 epochs)	27.6	[14.0, 56.2]	53 min
	SelfSup-Calib (350 epochs)	24.1	[11.1, 47.4]	6 h 11 min
	SelfSup-Calib (full dataset)	6.48	[–, –]	8 h 19 min
TUM	SelfSup-Calib (50 epochs)	29.7	[17.6, 44.5]	25 min
	SelfSup-Calib (350 epochs)	24.7	[16.2 , 63.6]	3 h 35 min
	SelfSup-Calib (full dataset)	4.31	[–, –]	2 h 51min
EuRoC raw	SelfSup-Calib (50 epochs)	10.8	[1.63, 47.9]	53 min
	SelfSup-Calib (350 epochs)	10.7	[1.81 , 55.9]	6 h 05 min
	SelfSup-Calib (full dataset)	2.24	[–, –]	8 h 27min

D Additional experimental results

This chapter contains additional experimental results, complementing the experiments shown in the main thesis.

D.1 Comparison of bias metrics on real lenses

In Chap. 3, the bias ratio [Hag20c, Hag22b] is introduced and compared to different bias metrics based on simulated calibration datasets. Following [Hag22b], this section extends this comparison to real lenses.

Using the calibration datasets described in Sec. 3.3.1 (see also Tab. D.1), we performed calibrations using models of different complexity and evaluated all bias metrics, including the root mean squared reprojection error (RMSE), the bias ratio (BR), the Kullback Leibler divergence (KLD) [Sch20], and the residual histogram [Bec18].

Fig. D.1 shows that across all three lenses, the bias ratio indicates the bias for insufficiently complex models (C(8) and C(3)). The residual histogram also provides a clear indication, but comes with the disadvantage of requiring visual inspection. The KLD does not always provide a clear indication (see low value for lens L3, model C(3)), and the RMSE increases with increasing bias, however, as explained in Chap. 3, its absolute values are generally difficult to compare between calibrations with different measurement noise.

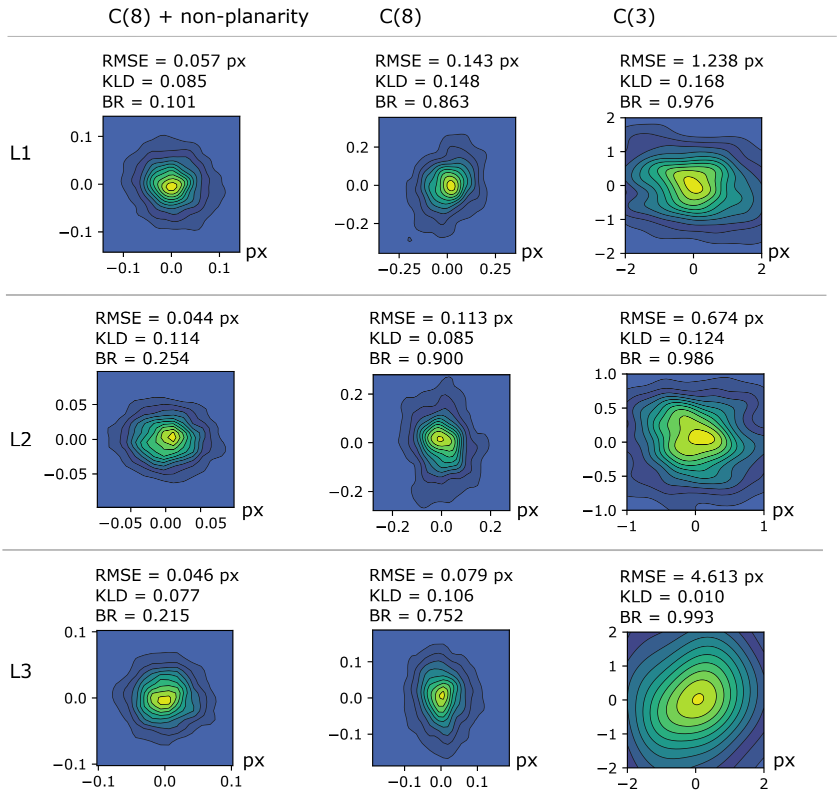


Figure D.1: Comparison of bias metrics on real lenses L1, L2, L3. The columns show calibrations with different models, including a pinhole model C(3), the fisheye model C(8), and the fisheye model plus a non-planar target model (Sec. 2.3.4). The compared metrics are the root mean squared reprojection error (RMSE), the Kullback-Leibler divergence (KLD), the bias ratio (BR), and the histogram of residuals (see Sec. 3.1 for details). Figure adapted from [Hag22b].

Table D.1: Reference intrinsics of the different calibration datasets.

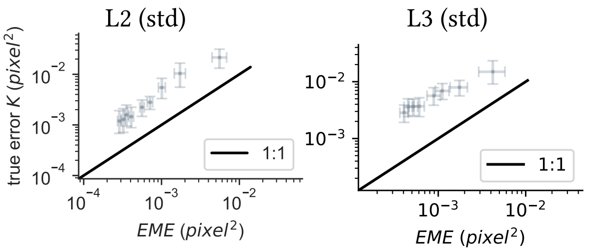
	f_x	f_y	c_x	c_y	κ_1	κ_2	κ_3	κ_4
simulation	4000	4100	2000	2000	-0.1	0.09		
rendering	900	900	361	361	-0.4	-0.2		
L1	3000.9	3001.3	970.3	614.1	-0.23	0.53		
L2	2163.8	2164.0	944.9	597.6	-0.10	0.17		
L3	851.9	852.0	947.6	601.9	0.06	0.06	-0.04	0.05

D.2 Validation of uncertainty estimation on real lenses

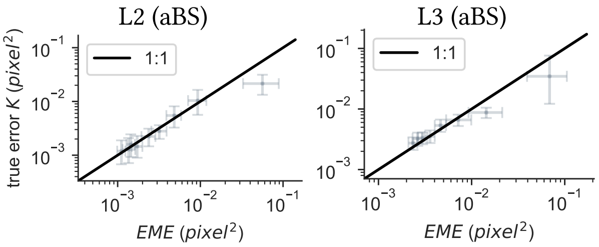
To validate the prediction of the expected mapping error (EME) derived in Chap. 4, Fig. D.2 shows the true mapping error K of different calibrations, plotted against the EME, as adapted from [Hag22b].

Consistent with the findings of Sec. 4.2.2, the EME significantly underestimates the true mapping error if it is computed based on the standard covariance estimator (Fig. D.2 a). However, if computed based on the approximated bootstrap (aBS) covariance estimator, the EME coincides well with the average true mapping error (Fig. D.2 b).

Fig. D.2 b further shows a limitation of the bootstrapping approach [Hag22b]: For calibration with few images ($N_{\mathcal{J}} \leq 15$), the uncertainty estimate becomes less accurate and the EME overestimates the true mapping error. This can be explained by the fact that the theory behind bootstrapping relies on the assumption of infinite sample sizes [Dav97]. Although bootstrapping is commonly used for finite sample sizes, the results become less accurate with decreasing sample size. Overall, Fig. D.2 shows that the EME consistently predicts the true mapping error, if computed based on an accurate estimate of the covariance matrix.



(a) standard



(b) approximated bootstrap

Figure D.2: Validation of the expected mapping error (EME) on calibrations of two real lenses (L2 and L3). When computed based on the standard covariance estimator (std), the EME increases with the true error, however, the absolute values of the EME underestimate the true error. When computed based on the aBS method, the magnitude of the EME is vastly consistent with the average true error. Only for small dataset sizes, the EME overestimates the true error. The data points show calibrations with different numbers of images ($N_{\mathcal{J}} \in \{10, 15, ...50\}$). Error bars are 95% bootstrap confidence intervals across 50 random samples for each $N_{\mathcal{J}}$. Figure adapted from [Hag22b].

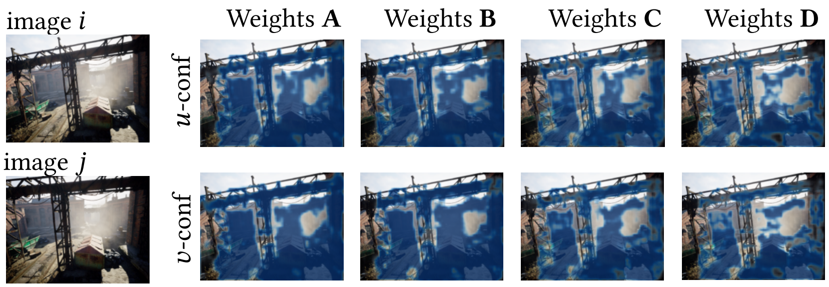


Figure D.3: Qualitative comparison of different training setups. Exemplary confidence weights $\mathbf{w}_{ij}$ predicted in the last update iteration k using the learned weights **A**-**D** obtained with different training setups. The blue overlay decreases with an increasing confidence weight, using a fixed linear scaling in the range $[0, 1]$.

D.3 Effect of different training setups on self-calibration

To analyze the effect of different training setups on the self-calibration introduced in Chap. 6, this section compares the learned weights obtained with four different settings, as also shown in [Hag23][1]:

1 **Weights A** are the learned weights obtained with the training setup described in Sec. 6.3.2, using the intrinsics loss $\mathcal{L}_\theta$ and exposing the model to initial intrinsics errors $\boldsymbol{\Delta\theta}$ during training.

2 **Weights B** are obtained with the same setup as **A**, except that the explicit intrinsics loss $\mathcal{L}_\theta$ is not included during training.

3 **Weights C** are obtained with the same setup as **B**, but without exposing the model to initial intrinsics errors $\boldsymbol{\Delta\theta}$ during training.

4 **Weights D** are the pre-trained weights from DROID-SLAM [Tee21], obtained through training with the original bundle adjustment layer that assumes accurately known intrinsics.

[1] Note that we here refer to the *learned weights*, i.e. the learned parameters of the deep neural network, not to the *confidence weights* that are predicted by the model.

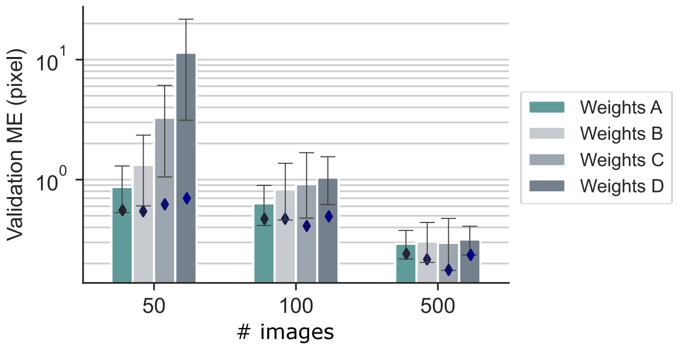


Figure D.4: Calibration accuracy of all four training setups for different sequence lengths. Results are obtained on random sequence snippets from the TartanAir validation sequences[a], using five random snippets per sequence. The barplot shows the average mapping error and 95% bootstrap confidence intervals, as well as the median mapping error (diamond) across all sequence snippets. Figure adapted from [Hag23].

[a] Sequences neighborhood/Easy/P021, abandonedfactory/Hard/P011, office/Hard/P007, westerndesert/Easy/P013, japanesealley/Easy/P007, gascola/Hard/P009.

To compare weights **A**-**D**, they are loaded as DroidCalib weights at inference time and employed with SC-BA layer. As a first qualitative observation, it can be seen that the learned weights **A** obtained through training with SC-BA layer produce, on average, larger and less spatially constrained confidence weights than the pre-trained weights **D** from [Tee21] (Fig. D.3).

For a quantitative evaluation, we use multiple validation sequences from the TartanAir dataset and evaluate the accuracy of estimated intrinsics. For all evaluations, we use naive initial intrinsics $\boldsymbol{\theta}_0$ (6.12) and quantify self-calibration accuracy in terms of the mapping error (ME) with respect to the ground truth or reference intrinsics.

Fig. D.4 shows that for long sequences ($N_\mathcal{J} = 500$), calibration accuracy does not differ significantly between weights **A**-**D**. Even the pre-trained weights from [Tee21] (weights **D**) result in comparatively high calibration accuracy when combined with the SC-BA layer at inference time. This suggests that camera pose estimation and self-calibration impose similar requirements on

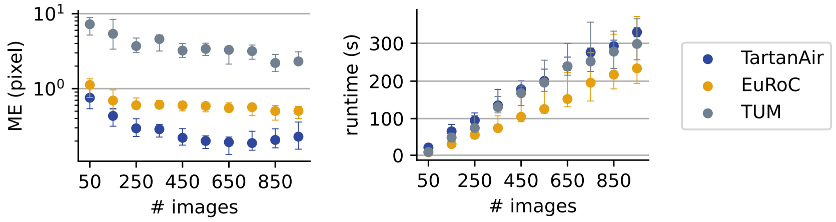


Figure D.5: Calibration accuracy (mapping error ME) and runtime of the proposed self-calibration approach depending on the number of input images. Plots show median values and 95% bootstrap confidence intervals on random sequence snippets of different lengths, with $\Delta_\theta = 25\%$, and three random snippets per sequence. Values are obtained by running a full estimation (frontend and backend) on a single Nvidia Titan RTX 24 GB GPU. Figure adapted from [Hag23].

the model, which is plausible, as both tasks require accurate correspondences on the static environment.

For short sequences ($N_\mathcal{J} = 50$), however, weights (**A**) result in higher calibration accuracy than the pre-trained DROID-SLAM weights (**D**). In particular, the *average* mapping error differs, while the effect on *median* mapping error is smaller. This suggests that for short sequences, the dedicated self-calibration training results in fewer gross errors, i.e. more robust self-calibration.

[Tee21] additionally evaluates how the DROID-SLAM system compares to a composed system that uses the pre-trained RAFT [Tee20] model for optical flow, followed by a bundle adjustment layer (i.e. without joint end-to-end training and without confidence weights). The pose estimation accuracy of the composed system was found to be significantly lower which suggests that end-to-end training and confidence weights are key components for accurate estimation [Tee21]. As the introduced self-calibration adopts the DROID-SLAM architecture, this relevance of confidence weights and end-to-end training can be expected to apply to self-calibration as well.

D.4 Self-calibration evaluation depending on sequence length

Self-calibration accuracy, as well as the required computational resources, depend on the length of the image sequence. To analyze the behavior of the self-calibration approach introduced in Chap. 6, we applied it on random sequence snippets of different lengths (Fig. D.5), as also shown in [Hag23].

In general, the computation time of DroidCalib is higher than that of DROID-SLAM (frontend running at 4 fps on TartanAir as compared to 10 fps for DROID-SLAM) which can be explained by the additional dense matrix operations required when estimating the intrinsics (Fig. 6.4).

Furthermore, there is a trade-off between calibration accuracy and required computational resources (computation time and memory consumption). For sequences from the TartanAir and EuRoC dataset, Fig. D.5 suggests that around 200 images are sufficient to obtain a mapping error below 1 pixel. For such short sequences, the average computation time is below two minutes. For the more challenging TUM sequences, however, even 900 images are not sufficient to achieve an average mapping error below 1 pixel. This suggests that for self-calibration, the quality of a sequence, including diversity of camera motion, amount of artifacts, and informative structure, is more relevant than the quantity of images [Hag23].