Diploma Thesis

Calibration of a Multi-Camera Rig From Non-Overlapping Views

Sandro Esquivel Olmos

Kiel, 17th September 2007

Abstract:

This work describes a novel approach to estimate the relative poses of multiple cameras fixed inside a rig. In contrast to existing methods for stereo and multi-camera rig calibration, overlapping views of the individual cameras are not required.
The proposed approach estimates the rig parameters, i.e. relative position and orientation of the fixed cameras with respect to each other, using time-corresponding poses for each camera during an arbitrary motion of the rig. This estimation can be done by solving systems of linear equations. The corresponding poses are previously obtained using pose estimation techniques from a sequence of time-corresponding images for each camera individually. Afterwards the rig parameter estimates are refined by a non-linear optimization.
It is shown that the presented calibration method is comparable in accuracy and efficiency to common rig calibration techniques that require overlapping views.

Diplomarbeit

Kalibrierung von starr gekoppelten Mehrkamerasystemen mit nicht-überlappenden Sichtbereichen

Sandro Esquivel Olmos

Kiel, 17. September 2007

Zusammenfassung:

Die vorliegende Arbeit beschreibt eine neue Methode, um die relativen Posen von Kameras in einem starr gekoppelten Mehrkamerasystem (Rig) aus Kamerabildern zu schätzen. Im Gegensatz zu bestehenden Verfahren zur Kalibrierung von Stereo- und Mehrkamerasystemen setzt der vorgestellte Ansatz keine gemeinsamen Bildbereiche der einzelnen Kameras voraus.
Dabei werden die Rig-Parameter, also relative Lage und Orientierung der gekoppelten Kameras zueinander, aus zeitlich korrespondierenden Posen der einzelnen Kameras während einer beliebigen Bewegung des Kamerasystems geschätzt. Diese Schätzung benötigt lediglich das Lösen linearer Gleichungssysteme. Die korrespondierenden Posen werden dabei aus einem vorangehenden Poseschätzverfahren aus einer Sequenz von zeitlich korrespondierenden Bilder für jede Kamera separat ermittelt. Danach wird die Schätzung durch weitere nichtlineare Optimierung verbessert.
Es wird gezeigt, dass der vorgestellte Ansatz bezüglich Genauigkeit und Effizienz vergleichbar gute Ergebnisse wie gängige Verfahren zur Kalibrierung von Mehrkamerasystemen mit gemeinsamen Sichtbereichen liefert.

Contents

Abbreviations and Symbols

$\mathcal{J}$	Image with width $w_\mathcal{J}$ and height $h_\mathcal{J}$ in pixels
$\mathbb{R}^2, \mathbb{P}^2$	Euclidean/projective 2d space
$\mathbb{R}^3, \mathbb{P}^3$	Euclidean/projective 3d space
$\tilde{\mathrm{m}} = (u, v)^\mathsf{T}$	2d image point in pixels
$\mathrm{m} = (x, y[, w])^\mathsf{T}$	2d image point (Euclidean/projective)
$\mathrm{M} = (X, Y, Z[, W])^\mathsf{T}$	3d space point (Euclidean/projective)
$\dot{\mathrm{m}} = (\varphi, r)$	Polar coordinates of 2d image point
$\dot{\mathrm{M}} = (\Phi, \Theta, R)$	Polar coordinates of 3d space point
K	Camera calibration matrix or function (intrinsic parameters)
A	General linear transformation matrix
$\mathrm{A}_j, \mathrm{a}_i$	j-th row or i-th column vector of matrix A
$\mathrm{A}_{(j,i)}$	Lower right submatrix of A beginning at j-th row and i-th column
a_{ji}	Entry at j-th row and i-th column of matrix A
$\mathbf{T} = [\mathrm{A} \mid \mathrm{b}]$	General affine transformation
$\mathbf{T} = [\lambda\mathrm{A} \mid \mathrm{b}]$	General similarity transformation
C	Euclidean coordinates of camera center
$[\mathrm{R} \mid \mathrm{C}]$	Euclidean transformation with translation C and rotation R, or camera pose with position C and orientation R (extrinsic parameters)
$\mathbf{P} = \mathrm{K}(\mathrm{R}^\mathsf{T} \ -\mathrm{R}^\mathsf{T}\mathrm{C})$	Camera projection matrix or function
F	Fundamental matrix
E	Essential matrix
$\mathrm{R}_{\mathrm{r},\alpha}$	Rotation around axis r by angle α
$\mathrm{R}_\mathbf{q}$	Rotation corresponding to unit quaternion $\mathbf{q}$
$\mathbf{q} = \langle q, \mathrm{q} \rangle$	Quaternion with scalar part q and vector part $\mathrm{q} = (x, y, z)^\mathsf{T}$
$\mathbf{T}_\mathbf{q}$, $\mathbf{T}^*_\mathbf{q}$	Left and right multiplication matrix of quaternion $\mathbf{q}$
$i = 0, \ldots, N$	Index of camera in rig (0: master camera, 1 to N: slave cameras)
$k = 0, \ldots, K$	Index of time step or image frame (0: initial frame)
$[\mathrm{R}_i^k \mid \mathrm{C}_i^k]$	Pose transformation of camera i at time step k
$[\Delta\lambda_i\Delta\mathrm{R}_i \mid \Delta\mathrm{C}_i]$	Rig parameters of slave camera i (internal pose transformation)

1. Introduction

Motivation

There is a variety of applications for rigidly coupled cameras with non-overlapping fields of view: Robots mounted with antipodal wide-angle cameras are used for sewer inspection. In the automotive industry information from non-overlapping front and rear view cameras is utilized for environment analysis. Multi-camera rigs encompassing a combined field of view of almost 360° are of rising interest in tasks varying from video surveillance and monitoring systems to 3d reconstruction of urban architecture. A well-known example for a multi-camera rig used for scene reconstruction is e.g. the Point Grey Ladybug®[1]. The main advantage of such devices with respect to conventional imaging systems is that visual information from a large field of view can be assembled without requiring expensive omni-directional cameras. Camera rigs have also considerably higher resolution than available omni-directional cameras. Furthermore, it has been shown that 3d scene reconstruction can be done very efficiently using a rig with a large field of view [FKK04, Boe07].
Applications using multiple camera rigs demand for an accurate calibration of the device which includes intrinsic parameter measurement and identification of the relative poses of the coupled camera with respect to each other. While there are numerous established solutions for pose calibration of multi-camera rigs with overlapping camera views, especially for stereo rigs, there are few such solutions present for rigidly coupled cameras *without* overlapping views. The main issue of this work is to investigate the pose calibration problem for multi-camera rigs without assuming overlapping fields of view based on computer vision techniques.

Our Approach

For this diploma thesis we developed a novel method to calibrate a multi-camera rig from images *without* overlapping views - from theoretical design to implementation,

[1] See Point Grey Research website at http://www.ptgrey.com for further details.

practical application, and evaluation. Our approach uses methods from the field of hand-eye calibration and estimates the rig parameters using time-corresponding poses of all cameras each in their respective local coordinate frame. The estimation approach does not depend on point correspondences between different cameras. Time-corresponding poses are provided by an image-based pose estimation method from synchronously captured camera image sequences.

Structure of this Work

This work is structured as follows. In **Chapter 2** we will at first refer to previous work done in the field of this topic and motivate our approach.
Chapter 3 describes the theoretical background needed to understand the issue. First, basic topics such as coordinate systems and rigid transformations, projections and camera models, and multiple view geometry are described. Then methods for intrinsic camera calibration and the pose estimation framework used to provide time-corresponding poses for practical applications are described. Finally, an overview over the main numerical approaches and algorithms that are performed in order to solve the arising problems is given.
The main topic of this work, i.e. estimation of the internal poses of each camera inside the rig from time-corresponding poses, is covered in **Chapter 4**. We will specify the main problem in an accurate way according to the models presented in the previous chapter, examine the condition of our problem formulation, and present a framework for solving the calibration task considering different settings.
In **Chapter 5** we will refer to our implementation, present its results in test cases and real applications and evaluate its stability and performance.
The results of this work will be resumed and concluded in **Chapter 6**.
Appendix I deepens mathematical and theoretical topics that are mentioned during the work which exceed the limitations of the main part of the diploma thesis.
In **Appendix II** a short reference to our implementation is given for practical applications.

Motivation

Im Umfeld der Computergrafik haben sich in den letzten Jahren zahlreiche Anwendungsbereiche für starr gekoppelte Mehrkamerasysteme ohne überlappende Sichtbereiche etabliert: Im Bereich der Kanalinspektionssysteme werden bereits Inspektionsfahrzeuge eingesetzt, die mit einander gegenüberliegenden Weitwinkelkameras bestückt sind. In der Automobilindustrie werden nicht-überlappende Front- und Heckkameras zur Umgebungsanalyse verwendet. Mehrkamerasysteme, die ein kombiniertes Sichtfeld von bis zu $360°$ abdecken, sind darüberhinaus von wachsendem Interesse für verschiedene Aufgaben, von Videoüberwachung und Kontrollsystemen bis hin zur 3D-Rekonstruktion von Stadtszenen. Ein Beispiel für ein am Markt etabliertes Mehrkamerasystem ist etwa der Ladybug® von Point Grey[2]. Der Vorteil solcher Systeme gegenüber herkömmlichen Kamerasystemen liegt in der Bereitstellung von visuellen Informationen über einen großen Sichtbereich, wobei Mehrkamerasysteme im allgemeinen günstiger sind als spezielle omnidirektionale Kameras.
Für Anwendungen mit starr gekoppelten Mehrkamerasystemen ist eine präzise Kalibrierung notwendig, die neben der Bestimmung der intrinsischen Kameraparameter das Ermitteln der relativen Posen der gekoppelten Kameras relativ zueinander umfasst. Während für die Kalibrierung von Mehrkamerasystemen mit gemeinsamen Sichtbereichen zahlreiche Verfahren vorliegen, insbesondere für Stereosysteme, liegen nur wenige Arbeiten vor, die sich mit Kamerasystemen *ohne* überlappende Sichtbereiche befassen. Die Problemstellung der vorliegenden Diplomarbeit besteht darin, ein Verfahren zur Kalibrierung eines starr gekoppelten Mehrkamerasystems zu formulieren, welches keine gemeinsamen Sichtbereiche der einzelnen Kameras voraussetzt. Das zu entwickelnde Verfahren soll dabei auf gängige Methoden aus dem Bereich der Computergrafik und Bildverarbeitung zurückgreifen.

Eigener Ansatz

In dieser Diplomarbeit wird ein neues Verfahren zur Rig-Kalibrierung aus nicht-überlappenden Kamerabildern erarbeitet - theoretisch entworfen, implementiert und praktisch getestet. Dieses Verfahren greift auf Methoden der Hand-Eye-Kalibrierung zurück und schätzt die Rig-Parameter aus zeitkorrespondierenden Posen aller Kameras in ihrem jeweils lokalen Koordinatenbezugssystem. Zeitkorrespondierende Posen werden hierbei durch ein bildbasiertes Poseschätzverfahren aus synchron aufgenommenen Bildsequenzen der Kameras zur Verfügung gestellt.

[2]Für weitere Details siehe Internetpräsenz von Point Grey Research unter http://www.ptgrey.com.

Gliederung der Arbeit

In **Kapitel 2** soll zunächst ein Überblick über bisherige Arbeiten und Ergebnisse im Umfeld des behandelten Themas gegeben werden und unser Ansatz motiviert werden.
In **Kapitel 3** wird der theoretische Hintergrund erläutert, der zum Verständnis der vorliegenden Arbeit notwendig ist, und die verwendetene Notationen eingeführt. Als erstes werden Grundlagenelemente der Computergrafik vorgestellt, einschließlich Koordinatensystemen und starren Transformationen, Projektionen und Kameramodellen, sowie die Geometrie multipler Kamerasichten. Danach werden die in den praktischen Problemstellungen eingesetzten Verfahren zur intrinsischen Kamerakalibrierung und Poseschätzung zur Bereitstellung zeitlich korrespondierender Posen aus Kamerabildern erläutert. Abschließend wird ein Überblick über die numerischen Methoden und Algorithmen gegeben, die zur praktischen Lösung der vorliegenden Probleme verwendet werden.
Das Hauptthema wird in **Kapitel 4** bearbeitet. Zunächst wird das vorliegende Problem in Hinsicht auf die im vorigen Kapitel vorgestellten Modelle spezifiziert, die Kondition des formulierten Problems untersucht und ein Framework zum praktischen Lösen der Kalibrierungsaufgabe unter verschiedenen Voraussetzungen vorgestellt.
In **Kapitel 5** wird die Implementierung des Lösungsansatzes ausgewertet und hinsichtlich Performanz und Stabilität in theoretischen Testfällen und praktischen Anwendungen untersucht.
Die Ergebnisse der Arbeit werden in **Kapitel 6** zusammengefasst und bewertet.
Anhang I vertieft theoretische und mathematische Themen, die im Laufe der Arbeit angesprochen werden, deren tieferes Verständnis aber über den Rahmen des für die vorliegende Arbeit Notwendigen hinausgeht.
Anhang II stellt eine kurze Referenz zur Implementierung für die praktische Umsetzung dar.

2. Previous Work

The calibration of multi-camera rigs, especially stereo systems, is an extensively researched problem in computer vision, e.g. to use the rig to recover metric structure from corresponding images. For this task both the intrinsics parameters of each camera and the relative poses - position and orientation - of each camera inside the rig have to be known. Traditional approaches assume that the cameras have spatial overlap between their fields of view and use the overlapping image parts to register the internal cameras with each other as suggested e.g. in the seminal work by *Zhang et al.* on perspective stereo rig calibration [ZLF96]. These approaches have in common that certain points must be visible for all cameras at the same time and internal poses of the rig cameras are estimated from the corresponding epipolar geometry

For the case of rig calibration from non-overlapping views there have been proposed few approaches which impose additional restrictions. Important contributions to this issue have been made by *Caspi and Irani* who proposed a technique to register images from a multi-camera rig without overlapping views with each other [CI01] but subject to the constraint that either all cameras share approximately the same center of view or the joint translation of the cameras is negligible relative to the distance of the scene.

To approach general configurations it appears more appropriate to relate the problem of rig calibration to the problem of *hand-eye calibration*:
Whenever a sensor is mounted onto a movable device - such as e.g. a camera mounted onto a robot hand or a camera with a gyroscope attached - and measurements from the sensor are to be mapped into the device's workspace frame, the relationship between sensor and device needs to be known. The problem of determining this relationship - most commonly described by a rigid 3d transformation X - from corresponding transformations A of the sensor and B of the device is referred to as *hand-eye calibration*. Figure 2.1 illustrates a typical hand-eye calibration setup. The cameras of the rig can be thought of as different sensors ("eyes" and "hands") in the hand-eye calibration model as illustrated in fig. 2.2, while the device workspace frame corresponds to the local coordinate frame of the rig.

There has been extensive research in the field of hand-eye calibration especially for the particular case of the sensor being a camera. Almost all proposed solutions demand for the solution of a homogeneous matrix equation of the form $\mathrm{AX} = \mathrm{XB}$ where A and B

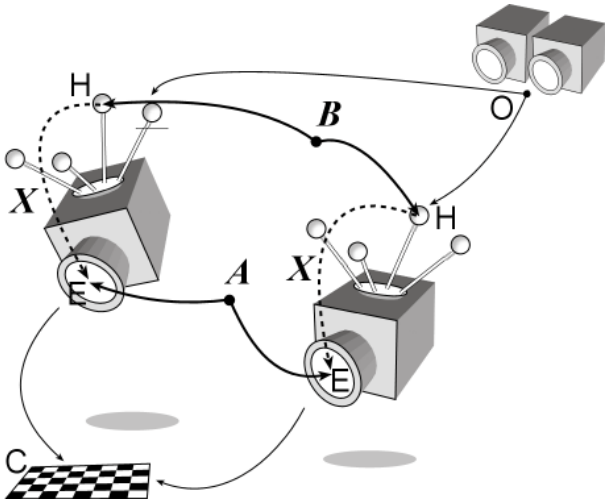


Figure 2.1.: Common hand-eye calibration problem for a camera E mounted with a robot hand H. A denotes pose transformations of the eye, B of the hand, and X is the relative pose transformation between hand and eye. Here, relative pose transformations of the hand are measured by a fixed observation system O, and relative pose transformations of the eye are measured with respect to a fixed calibration object C.

are related with time-corresponding pose transformations of sensor and device while X is related to the relative pose of the sensor with respect to the device which is fixed over time. Pose transformations A and B are thereby provided by methods specific to the sensors used.
Early solutions regard the rotational part of this equation decoupled from the translational part resulting in simple, linear formulations. Such approaches were proposed first by *Shiu and Ahmad* [SA89] (who performed a least squares estimation first of rotation first, then of translation, using the angle-axis representation for orientations) and *Tsai and Lenz* [TL89] (similar to [SA89] using a closed-form solution) for sensors mounted onto robots. *Wang* compared both methods resulting in a slight advantage for the latter [Wan92]. Similar approaches using different pose representations where proposed by *Chen* using screw motion representation [Che91] (the first approach that does not decouple the rotational and translational equations parts) and *Chou and Kamel* using quaternions for orientation representation [CK91] that make use of the singular value decomposition (SVD) for solution.
While all these approaches consider linear optimization techniques, *Horaud and Dornaika* proposed a non-linear optimization for the rotational and translational parts one-to-one [HD95] yielding more robustness with respect to sensor noise and measurement errors.

Our approach interprets the problem of rig calibration in a similar way as the problem of hand-eye calibration which will be modeled in **Chapter 4**. As we will point out, the

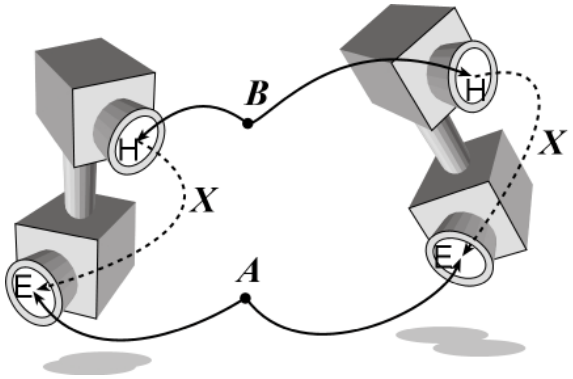


Figure 2.2.: Stereo-rig calibration interpreted as hand-eye calibration problem.

considered problem of rig calibration differs from the hand-eye calibration problem by the fact that additional unknown scale parameters have to be introduced. We will use linear estimation and non-linear refinement methods similar to [CK91] and [HD95] to approach the problem. Based on the theoretical solution strategies we will develop a framework for rig calibration and evaluate an implementation in **Chapter 5**.
At first, **Chapter 3** presents the theoretical background for addressing the problem in detail, starting with the basic topics of geometry, coordinate systems, and transformations in order to understand camera models, multiview geometry, and finally pose estimation techniques.

3. Theoretical Background

3.1. Geometry, Coordinate Systems and Transformation

To measure geometric entities and to describe geometric properties, the image plane or space is identified with an appropriate vector space provided with a coordinate system which is chosen specifically to the given problem. Transformations of entities describing processes such as rigid motions, projections, or changes of the coordinate system, can thereby be expressed in terms of linear algebra. In this section Euclidean and projective geometry are introduced which provide a proper geometric and algebraic model for computer vision, and common transformations are defined. Then we will describe coordinate systems and changes of reference coordinate frame appropriately for the task of rig calibration in the sense of hand-eye calibration.

3.1.1. Projective Geometry and Transformations

In this section we will introduce projective geometry and transformations that are needed to describe the projections of perspective cameras. We will begin with projective geometry for the planar space and lead over to the 3d space.

Projective geometry defines an extension of Euclidean geometry. While in Euclidean geometry of the 2d image plane the parallel axiom holds, i.e. from a geometrical view there are lines which do not intersect, projective geometry is altered such that all lines intersect. This is performed by adding *points at infinity* such that parallel lines intersect each in a point at infinity which corresponds to their direction. The set of all points at infinity forms the *line at infinity*.

The projective plane: To describe points and lines in the perspective plane, the *homogeneous* notation for points and lines in the plane is introduced. Each point of the

Euclidean plane may be represented by a pair of coordinates $\mathrm{m} = (x, y)^\mathsf{T} \in \mathbb{R}^2$ while lines are represented by a triplet $\mathbf{l} = (a, b, c)^\mathsf{T} \in \mathbb{R}^3$ such that all points $(x, y)^\mathsf{T} \in \mathbb{R}^2$ satisfying $ax + by + c = 0$ lie on the line. Since for a fixed $(a, b, c)^\mathsf{T} \in \mathbb{R}^3$, all triplets $k(a, b, c)^\mathsf{T}$ represent the same line for any non-zero $k \in \mathbb{R}$, vectors $(a, b, c)^\mathsf{T} \in \mathbb{R}^2$ which are related by a non-zero scale are considered as equivalent. Each equivalence class of vectors due to this relationship is denotes as a homogeneous vector. The set of all equivalence classes form the projective space $\mathbb{P}^2$ which can be identified with $\mathbb{R}^3 \setminus (0, 0, 0)^\mathsf{T}$.
A point $(x, y)^\mathsf{T}$ in $\mathbb{R}^2$ is represented in $\mathbb{P}^2$ by the homogeneous coordinates $k(x, y, 1)$ for any non-zero $k \in \mathbb{R}$. Given a homogeneous vector $\bar{\mathrm{m}} = (x, y, w)$, $w \neq 0$, the Euclidean representative can be obtained from $\mathrm{m} = (x/w, y/w)$. Homogeneous points $\bar{\mathrm{m}} = (x, y, w)^\mathsf{T}$ with $w \neq 0$ correspond to finite points which can be found in the Euclidean plane while homogeneous points with $w = 0$ define the points at infinity. Equivalently the line at infinity is represented by the homogeneous vector $\mathbf{l}_\infty = (0, 0, 1)^\mathsf{T}$.
Interpreting $\mathbb{P}^2$ as $\mathbb{R}^3$ for clearness, each point in the Euclidean plane $\mathbb{R}^2$ equals a ray through the origin in $\mathbb{P}^2$ and vice versa, while lines in $\mathbb{R}^2$ correspond to planes in $\mathbb{P}^2$, and the Euclidean plane is embedded into $\mathbb{P}^2$ at $w = 1$.
Note that equality is defined *up to scale* for homogeneous vectors since $\bar{\mathrm{m}}$ equals $k\bar{\mathrm{m}}$ for any $k \neq 0$. This will be emphasized in this work by the use of the similarity symbol $\sim$ instead of the equality symbol $=$. Euclidean and homogeneous representation will be used interchangeably for finite points since the Euclidean representation can be obtained by normalization $\mathrm{m} = \frac{1}{w}\bar{\mathrm{m}}$ for $w \neq 0$.

Transformations of the projective plane: In this paragraph we will describe common transformations of the projective plane that can be described by linear transformations on homogeneous 3-vectors, represented by 3×3 matrices.

A **planar Euclidean transformation** or *isometry* is a transformation of inhomogeneous points that preserves Euclidean distance. It is given by the matrix representation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \mathrm{R} & \mathrm{C} \\ 0^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.1}$$

or, in short, $\bar{\mathrm{m}}' = [\mathrm{R} \mid \mathrm{C}]\bar{\mathrm{m}}$ where R is a 2×2 rotation matrix $\mathrm{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$ for an angle $\alpha \in [0, 2\pi)$ and $\mathrm{C} \in \mathbb{R}^2$ is a Euclidean 2-vector. Euclidean transformations model rigid motions of an object in the Euclidean plane, decomposed into a rotation R and additional translation by C. Hence a planar Euclidean transformation has 3 degrees of freedom and can be computed from two point correspondences.

The Euclidean transformation is generalized by the **similarity transformation** or *similarity* which does not preserve distance but shape. It is given by

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda \mathrm{R} & \mathrm{C} \\ 0^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.2}$$

or, in brief, $\bar{\mathrm{m}}' = [\lambda \mathrm{R} \mid \mathrm{C}]\bar{\mathrm{m}}$ where $\lambda \in \mathbb{R}$ represents an isotropic scaling. Similarity transformations model rigid motion and isotropic scaling and will be used to describe the change of coordinate systems. A planar similarity transformation has 4 degrees of freedom and can be computed from two point correspondences.

The most general transformation is the **planar projective transformation** which is also referred to as *homography*, defined on homogeneous points by

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{pmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} \tag{3.3}$$

abbreviated as $\bar{\mathrm{m}}' = \mathrm{H}\bar{\mathrm{m}}$. Note that H can be scaled by any non-zero scale factor without altering the projective transformation, i.e. such that eq. (3.3) holds for projective equality $\sim$. Hence a planar projective transformation has 8 degrees of freedom and can be computed up to scale from 4 point correspondences. Projective transformations leave only projective properties invariant, i.e. points are mapped to points, and lines are mapped to lines. Projective transformations are used to describe perspective projections of rays to an image plane as needed for common camera models as will be described in section 3.2.

The projective 3d space: The projective 3d space is created straightforward from the projective plane. The projective 3d space $\mathbb{P}^3$ consists of Euclidean 3d space extended by a set of points, lines, and the plane at infinity. The latter is equivalent to the line at infinity for the projective plane. In the context of this work we will only address points in the projective 3d space.
Each Euclidean point $\mathrm{M} = (X, Y, Z)^\mathsf{T} \in \mathbb{R}^3$ is represented by a homogeneous vector $\bar{\mathrm{M}} = (X, Y, Z, 1)^\mathsf{T}$. A homogeneous vector $\bar{\mathrm{M}} = (X, Y, Z, W)^\mathsf{T}$ with $W \neq 0$ corresponds to the Euclidean point $(X/W, Y/W, Z/W)^\mathsf{T}$. Homogeneous points with $W = 0$ represent points at infinity.

Transformations of the projective 3d space: Transformations of the projective 3d space are related directly to the transformation of the projective plane and may be described by linear functions on homogeneous vectors, represented by 4×4 matrices.

A 3d **Euclidean transformation** on inhomogeneous points $\mathrm{M} = (x, y, z)^\mathsf{T}$, $\mathrm{M}' = (x', y', z')^\mathsf{T}$ is given by

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \mathrm{R} & \mathrm{C} \\ 0^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.4}$$

or, in short, $\bar{\mathrm{M}}' = [\mathrm{R} \mid \mathrm{C}]\bar{\mathrm{M}}$ where R is a 3×3 rotation matrix and $\mathrm{C} \in \mathbb{R}^3$ is a Euclidean 3-vector. A Euclidean transformation models a rigid motion in 3d space consisting of rotation by R and translation by C, preserving distances and volumes. Since a 3d rotation has 3 degrees of freedom, a Euclidean transformation has 6 degrees of freedom and can be computed from 2 point correspondences.

A 3d **similarity transformation** is given by

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda\mathrm{R} & \mathrm{C} \\ 0^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.5}$$

or, in brief, $\bar{\mathrm{M}}' = [\lambda\mathrm{R} \mid \mathrm{C}]\bar{\mathrm{M}}$ where $\lambda \in \mathbb{R}$ defines the isotropic scale. A similarity transformation has hence 7 degrees of freedom and can be computed from 3 point correspondences.

Finally, a 3d **projective transformation** or homography is defined on homogeneous points by

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & & & \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} & & & \\ h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4}h_{4,1} & h_{4,2} & h_{4,3} & h_{4,4} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \tag{3.6}$$

or, more briefly, $\bar{\mathrm{M}}' = \mathrm{H}\bar{\mathrm{M}}$. Due to the mentioned scale ambiguity considering projective equality $\sim$, a projective 3d transformation of $\mathbb{P}^3$ has 15 degrees of freedom and requires 5 point correspondences to be computed.

3.1.2. Coordinate Systems

Coordinate systems are used to describe the positions of points in space. In an n-dimensional vector space the position of a point m is identified uniquely by the n-tuple $(x_1, \ldots, x_n)$ of its *coordinates* within the given coordinate system.

In computer vision it is appropriate to use *affine coordinate systems* to describe geometric objects in 2d and 3d. A real affine coordinate system is defined by an orthogonal base $\mathfrak{B} = \{\mathrm{b}_1, \ldots, \mathrm{b}_n\}$ of an n-dimensional real vector space V, and an origin vector $\mathrm{O} \in V$. It describes a mapping $\Phi_{\mathfrak{B},\mathrm{O}}$ of elements of V to n-tuples such that the inverse mapping is given by

$$\Phi^{-1}_{\mathfrak{B},\mathrm{O}} : \mathbb{R}^n \to V, \mathrm{x} \mapsto \sum_{i=1}^{n} x_i \mathrm{b}_i + \mathrm{O} \tag{3.7}$$

The canonical coordinate system is given by the canonical base $\mathfrak{B}_0 = \{\mathrm{e}_1, \ldots, \mathrm{e}_n\}$ and zero origin $\mathrm{O}_0 = (0, \ldots, 0)^\mathsf{T}$.

Geometrically interpreted, an affine coordinate system with an arbitrary orthogonal base $\mathfrak{B}$ and origin O can be interpreted as a *scaled*, *translated*, and *rotated* instance of the regarding canonical coordinate system. Moreover, given two affine coordinate system with orthogonal bases $\mathfrak{B}_1$, $\mathfrak{B}_2$ and origins O_1, O_2, the latter arises from the former by scaling, translating, and rotating, i.e. by a similarity transformation. This applies, because a rotation in V is defined by an orthonormal linear mapping which preserves length and orthogonality of vectors $v \in V$. Hence for two bases $\mathfrak{B}_1$, $\mathfrak{B}_2$ there exist a rotation matrix R and a diagonal scaling matrix Λ such that $\mathfrak{B}_2 = \{\Lambda \mathrm{R} \mathrm{b} | \mathrm{b} \in \mathfrak{B}_1\}$. The intermediate translation t is given by $\mathrm{O}_2 = \mathrm{O}_1 + \mathrm{t}$.

Throughout this work affine coordinate systems defined by an arbitrary base and origin will be denoted as *coordinate frames*. We will regard only coordinate frames that are scaled isometrically along all axes, i.e. there is a $\lambda \in \mathbb{R}$ such that $\Lambda = \lambda \mathrm{I}$. The canonical affine coordinate frame is considered as the *world coordinate frame* $\mathcal{C}^{world}$. Hence an affine coordinate frame $\mathfrak{B}, \mathrm{O}$ can also be uniquely parametrized by a rotation of the canonical base vectors defined by R, a translation of the zero origin denoted by C in world coordinates, and an isometrical scaling of the canonical base by a scale λ, holding:

$$\mathrm{C} = \Phi^{world}(\mathrm{O}) \qquad \text{and} \qquad \mathrm{b}_i = \lambda \mathrm{R} \mathrm{e}_i, \quad \text{for each } i = 1, \ldots, n \tag{3.8}$$

Rotation, translation, and scale of a coordinate frame with respect to the world coordinate frame are referred to as *absolute parameters* of the coordinate frame.
By fixing any *reference coordinate frame* $\mathcal{C}^{ref}$, described by the absolute parameters R^{ref}, C^{ref}, and λ^{ref}, any affine coordinate frame $\mathfrak{B}, \mathrm{O}$ can be furthermore uniquely parametrized with respect to $\mathcal{C}^{ref}$ by $\Delta\mathrm{R}$, $\Delta\mathrm{C}$, and $\Delta\lambda$ such that:

$$\Delta\mathrm{C} = \Phi^{ref}(\mathrm{O} - \mathrm{O}^{ref}) \qquad \text{and} \qquad \mathrm{b}_i = \Delta\lambda \Delta\mathrm{R} \mathrm{b}_i^{ref}, \quad \text{for each } i = 1, \ldots, n \tag{3.9}$$

$\Delta\mathrm{R}$, $\Delta\mathrm{C}$, and $\Delta\lambda$ are referred to as *relative parameters* of the coordinate frame with respect to its reference frame.

3.1.3. Rigid Transformations between Coordinate Frames

In the last section we pointed out how to parametrize general coordinate frames by the geometric terms rotation, translation, and isometric scale with respect to a reference coordinate frame or the world coordinate frame. Using this model for the projective 3d space it is easy to consider multiply nested coordinate frames, and to interpret coordinate frames geometrically as poses of cameras in the rig within the scene as described in the following section. Each coordinate frame can be characterized by its origin C, its orientation R and its isometric scale λ measured within the wrapping coordinate frame.

The change between two Cartesian coordinate frames of the Euclidean 3d space is described by a similarity transformation on inhomogeneous points. As defined in eq. (3.5), the similarity transformation is of the form

$$\mathbf{T} = \begin{pmatrix} \lambda\mathrm{R} & \mathrm{C} \\ 0^\intercal & 1 \end{pmatrix} \tag{3.10}$$

abbreviated as $\mathbf{T} = [\lambda\mathrm{R} \mid \mathrm{C}]$ where $\lambda \in \mathbb{R}$ accounts for the isometric scaling between the coordinate systems, R is a 3×3 rotation matrix describing the relative orientation of the reference frames to each other, and $\mathrm{C} \in \mathbb{R}^3$ describes the translation between the two reference frames or in other words: The origin of the transformed coordinate frame inside the untransformed coordinate frame (see fig. 3.1). Equivalently, each coordinate frame is identified with respect to a fixed reference frame by such a similarity transform.

When the scale λ is equal to 1, $\mathbf{T}$ is described by a Euclidean transformation as defined in eq. (3.4), abbreviated as $\mathbf{T} = [\mathrm{R} \mid \mathrm{C}]$:

$$\mathbf{T} = \begin{pmatrix} \mathrm{R} & \mathrm{C} \\ 0^\intercal & 1 \end{pmatrix} \tag{3.11}$$

This accounts geometrically for a rigid *motion* of a coordinate frame within its reference frame without change of scale.

The concatenation of two subsequent changes of reference frames $\mathbf{T}_1$ and $\mathbf{T}_2$ can then be computed by a simple matrix multiplication

$$\mathbf{T} = \mathbf{T}_2\mathbf{T}_1 \tag{3.12}$$

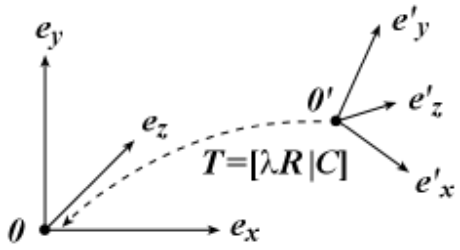


Figure 3.1.: Similarity transformation between coordinate frames

3.2. Camera Models

In general it is an essential issue of computer vision to use an appropriate mathematical model for the process of image formation. In this section we will describe the process of image generation with a monocular camera using the ideal *pinhole camera model* for perspective cameras. Also, the *spherical camera model* according to *Scaramuzza et al.* [SMS06] will be described to model wide-angle cameras. Aberrations from the ideal models that occur with real cameras due to lens distortion are compensated by *distortion models*. The issue of estimating the actual intrinsic camera parameters and distortion will be addressed in section 3.4. For a deeper insight into this topic we refer to [HZ00].

3.2.1. Perspective Camera Model

First, there are two basic types of camera projection models: the *perspective projection* model and the *orthographic projection* model. Perspective projection, also referred to as *central projection*, corresponds to the model of the *pinhole camera* where the visible scene is projected to the image via central projection with respect to a given projection center, the *camera center* (see fig. 3.2(a)).
Orthographic projection on the other hand is given by a special case of central projection where the projection center is infinitely distant from the scene (see fig. 3.2(b)). The visible scene is hence projected to the image plane neglecting depth differences in the scene. While perspective projection is a good model of the actual geometry of image formation with general cameras, orthographic projection yields more simple mathematics than perspective projection and is often used to approximate perspective projection. Seminal approaches using orthographic images for pose estimation have been proposed by e.g. *Tomasi and Kanade* [TK92]. Nevertheless, since its applicability is limited, we will focus on the perspective camera model in this work.

Without assuming radial distortion, an ideal *pinhole camera* provides an appropriate

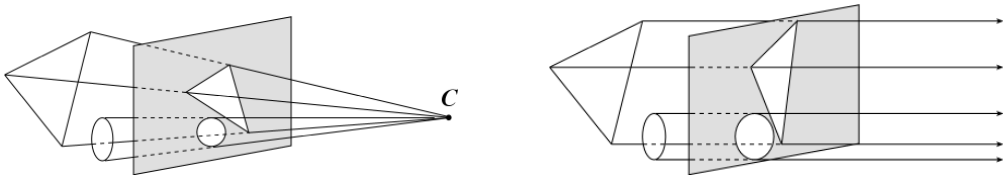


Figure 3.2.: Camera projection models (a) Perspective projection with central point C (b) Orthographic projection with infinitely distant projection center

model for a general CCD camera. The pinhole camera as shown in fig. 3.3 basically describes the central projection of points in 3d space onto a plane[1]. The image dimensions are given by width w and height h. The distance between central point and image plane is denoted as the *focal length* f. The line perpendicular to the image plane passing through the central point is denoted as the *optical axis*. Its intersection with the image plane is referred to as the *principal point* p of the camera image. Note that the projection center is referred to as the *camera center* C which defines the position of the camera in 3d space while the optical axis and image plane fix the orientation of the camera's coordinate frame: The image plane is by definition parallel to the x-y plane of the camera-local coordinate frame and located by convention at $z = 1$.

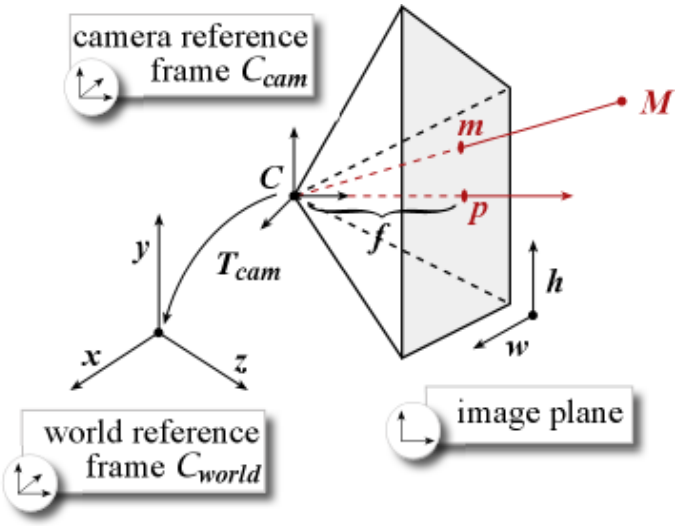


Figure 3.3.: The illustration shows the projection model of a perspective camera. 3d points M are projected to the image plane by means of central projection with respect to the camera center C. p denotes the principle point. The Euclidean transformation between camera reference frame and world coordinate frame is given by $\mathbf{T}_{cam}$.

[1]This notion refers to the original *camera obscura* where light rays from the visible scene pass through a very small pinhole into a dark box to form an image inside on the back wall where a light-sensitive film could be applied.

Using the pinhole camera model, the coordinates of a 3d point $\mathrm{M} = (X, Y, Z)^\mathsf{T}$ in the world camera system and its 2d pixel coordinates in the camera image $\tilde{\mathrm{m}} = (u, v)^\mathsf{T}$ are related by

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.13}$$

where $\mathbf{P}$ is a real 3×4 matrix referred to as the perspective *camera projection matrix* and $\sim$ indicates equality up to scale.
Denoting the homogeneous vector of $\tilde{\mathrm{m}}$ as $\bar{\mathrm{m}} = (u, v, 1)^\mathsf{T}$ and of M as $\bar{\mathrm{M}} = (X, Y, Z, 1)^\mathsf{T}$ eq. (3.13) can be written briefly as $\bar{\mathrm{m}} \sim \mathbf{P}\bar{\mathrm{M}}$.

The camera projection matrix $\mathbf{P}$ can be decomposed as

$$\mathbf{P} = \mathrm{K}(\mathrm{R}^\mathsf{T} \; -\mathrm{R}^\mathsf{T}\mathrm{C}) \tag{3.14}$$

where $\mathrm{K} \in \mathbb{R}^{3\times3}$ defines the *camera calibration matrix*

$$\mathrm{K} = \begin{pmatrix} fk_u & s & p_u \\ 0 & fk_v & p_v \\ 0 & 0 & 1 \end{pmatrix} \tag{3.15}$$

which is related to the *intrinsic parameters* of the camera, i.e.:

- The *focal length* f of the camera.
- Horizontal and vertical *pixel scale factors* k_u, k_v whose inverses define the size of each image pixel in the world coordinate unit. Common cameras are assumed to have square pixels, i.e. $k_u = k_v$.
- *Pixel skew* value s which accounts for non-orthogonality of the pixel grid. Pixels are customary considered to be rectangular, i.e. $s = 0$.
- The pixel coordinates p_u, p_v of the camera's principal point in the image, i.e. the intersection of the optical axis with the image plane measured in the image coordinate system.

Hence the calibration has five degrees of freedom given by fk_u, fk_v, s, p_u, and p_v. Under the assumptions $k_u = k_v$ and $s = 0$ - denoted as the *reduced camera model* which holds for the most currently manufactured cameras - the number of degrees of freedom is reduced to 3. Note that K is an upper triangle matrix and its inverse K^{-1} is defined.
Once K^{-1} is known, an image pixel $\tilde{\mathrm{m}} = (u, v)^\mathsf{T}$ can be back-projected into 3d space by:

$$(x, y, w)^\mathsf{T} \sim \mathrm{K}^{-1}(u, v, 1)^\mathsf{T} \tag{3.16}$$

These coordinates, commonly either scaled to length 1 ("normalized") or $w = 1$ ("homogenized"), will be referred to throughout this work as *normalized coordinates* $\bar{\mathrm{m}}_{norm}$ of $\tilde{\mathrm{m}}$. $\bar{\mathrm{m}}_{norm}$ can be visualized as the ray emitting from the camera center through an image point within the camera-local coordinate frame.

The 3×4 matrix $(-\mathrm{R}^\mathsf{T} \; -\mathrm{R}^\mathsf{T}\mathrm{C})$ accounts for the absolute *pose* of the camera within the world coordinate system and is related to as the *extrinsic parameters* of the camera, *orientation* R and *position* C. The 3×3 rotation matrix R is related to the physical orientation of the cameras and C to its position, i.e. the camera center, within the world coordinate system. This corresponds to the Euclidean transformation $[\mathrm{R} \mid \mathrm{C}]$ transforming the camera-local coordinate frame into the world coordinate frame, or to the inverse transformation $[\mathrm{R}^\mathsf{T} \mid -\mathrm{R}^\mathsf{T}\mathrm{C}]$ respectively. Note that the pose has 6 degrees of freedom, 3 accounting each for rotation and position.

Summing up, the projection matrix of a general projective camera has 11 degrees of freedom: 6 from the pose matrix, 5 from the calibration matrix. Since the overall scale of the perspective projection matrix can be ignored for projective reconstruction, the number of degrees of freedom can be reduced to 10. For the reduced camera model the number of degrees of freedom is further reduced to 8.

Throughout this work we assume that the pinhole camera model is an accurate model for the image formation process. In practice using real camera with light-focusing lenses, deviations from the linear model will occur. The most important deviation is in general a radial distortion which will be modeled as proposed in [HZ00, 7.4] by the radial distortion center $(c_u, c_v)^\mathsf{T}$ and a fourth-order polynomial L with coefficients $1, \kappa_1, \ldots, \kappa_4$ such that distorted image coordinates $\hat{\mathrm{m}} = (\hat{u}, \hat{v})$ are mapped to ideal coordinates $\tilde{\mathrm{m}} = (u, v)$ by an *undistortion function* τ defined by:

$$\tau(\hat{u}, \hat{v}) = (u, v) \qquad u = c_u + L(r)(\hat{u} - c_u) \qquad v = c_v + L(r)(\hat{v} - c_v) \tag{3.17}$$

where $r = \sqrt{(\hat{u} - c_u)^2 + (\hat{v} - c_v)^2}$ is the distance of distorted image pixels from the distortion center. The number of intrinsic parameters is hence increased by 6 and the mapping from normalized space points to image points is described by a non-linear mapping $\mathcal{K}$ instead of a linear mapping K.

3.2.2. Spherical Camera Model

To describe wide-angle or hemispherical cameras, or more specific: cameras with fish-eye lenses, we use the omni-directional camera model proposed by *Scaramuzza et al.* [SMS06]. In this model the image formation process can be thought of as projecting the visible scene onto the unit hemisphere around the camera center instead of the plane at $w = 1$, and addressing image pixels by means of their polar coordinates. This projection is commonly referred to as *angular fisheye projection*. It is illustrated in fig. 3.4.

Using the fisheye camera model, the coordinates of a 3d point $\bar{\mathrm{M}} = (X, Y, Z, 1)^\mathsf{T}$ in the world camera system and its 2d pixel coordinates in the camera image $\bar{\mathrm{m}} = (u, v, 1)^\mathsf{T}$ are related by a non-linear projection function $\bar{\mathrm{m}} = \mathcal{P}(\bar{\mathrm{M}})$ which can be decomposed as:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathcal{K}(\begin{pmatrix} x \\ y \\ z \end{pmatrix}), \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix} = (\mathrm{R}^\mathsf{T} \; -\mathrm{R}^\mathsf{T}\mathrm{C}) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.18}$$

where $\mathcal{K}$ denotes a non-linear mapping of normalized coordinates $(x, y, z)^\mathsf{T}$ first to spherical coordinates $(\Phi, \Theta, 1)^\mathsf{T}$ of the corresponding unit vector, then to Cartesian pixel coordinates $(u, v, 1)^\mathsf{T}$ with respect to the image center by:

$$\mathcal{K}(\begin{pmatrix} x \\ y \\ z \end{pmatrix}) = \begin{pmatrix} k_r & 0 & p_u \\ 0 & k_r & p_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Theta \cos \Phi \\ \Theta \sin \Phi \\ 1 \end{pmatrix} \tag{3.19}$$

$\mathcal{K}$ is denoted with respect to the camera calibration matrix as the *camera calibration function*. $(\mathrm{R}^\mathsf{T} \; -\mathrm{R}^\mathsf{T}\mathrm{C})$ denotes the transformation of points in the world coordinate frame to camera-local points as in the perspective camera model.

Note that the distance of a pixel from the center of the image, given by $r = \sqrt{(u - p_u)^2 + (v - p_v)^2} = k_r\Theta$, is proportional to the angle from the camera view direction Θ. Hence points seen under the same angle form circles on the image plane as shown in fig. 3.4. In particular, in an angular fisheye image the resolution is approximately equal across the whole image. Moreover, the linear mapping of the distance of image pixels to the image center to angles can be described by a single scale k_r if distortion is neglected.

In comparison with perspective cameras, the *intrinsic parameters* of an ideal fisheye camera are given by:

- The *pixel/radius ratio* k_r, i.e. the radius of the $1°$ equiangular circle in the camera image in pixels.
- The pixel coordinates p_u, p_v of the camera's principal point in the image, i.e. the image point corresponding to the polar coordinates $(0, 0, 1)^\mathsf{T}$ in the camera-local coordinate system.

Notice that the inverse mapping from image points $\tilde{\mathrm{m}}$ to normalized points $\bar{\mathrm{m}}_{norm}$ is given by:

$$\mathcal{K}^{-1}\left(\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}\right) = \begin{pmatrix} \frac{\sin\Theta}{r}(u - p_u) \\ \frac{\sin\Theta}{r}(v - p_v) \\ \cos\Theta \end{pmatrix} \sim \begin{pmatrix} u - p_u \\ v - p_v \\ r\cot\Theta \end{pmatrix} \tag{3.20}$$

where $r = \sqrt{(u - p_u)^2 + (v - p_v)^2}$ denotes the pixel distance from the image center, and $\Theta = \frac{r}{k_r}$.

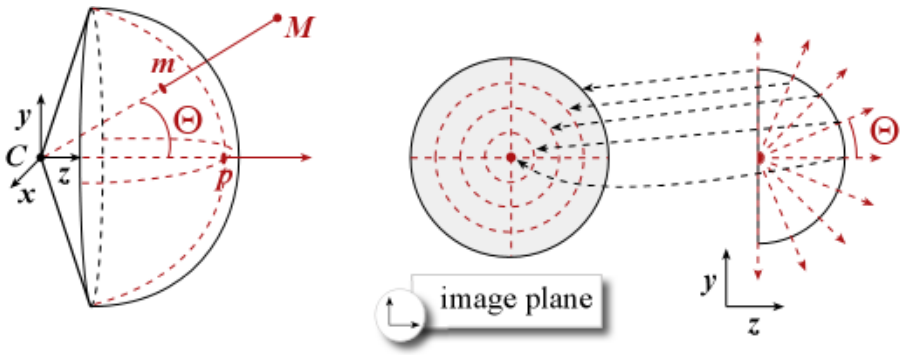


Figure 3.4.: Projection model of a spherical camera using angular fisheye projection. Illustration of a hemispherical camera with an aperture angle of $180°$ (left). The plotted lines in the circular image (center) denote the equiangular circles for fixed viewing angles of $\Theta = \frac{j}{4} \cdot 90°$ with $j = 0, \ldots, 4$ (right).

Distortion is modeled as a deviation of the viewing angle Θ and distance from image center r from being linearly correlated. [SMS06] suggests to describe the relationship between both parameters by a fourth-order polynomial $\Theta = \rho(r)$ with parameters $0, \kappa_1, \ldots, \kappa_4$ instead of $\Theta = r/k_r$.

Note that in applications where only normalized image points are considered we will not have to pay regard to the specific camera model being a perspective camera or a

spherical camera or to the distortion model, once the mapping $\mathcal{K}$ from camera-local coordinates to image pixels (and vice versa$\mathcal{K}^{-1}$ from image pixels to normalized coordinates) is known and fixed for a camera.

3.3. Epipolar Geometry and the Essential Matrix

The *epipolar geometry* defines the intrinsic projective geometry between two camera images depending only on the cameras' intrinsic parameters and relative pose. Epipolar geometry is exploited for a variety of problems such as depth estimation by triangulation, image point matching, and pose reconstruction from calibrated cameras. We will introduce epipolar geometry, the notion of the Essential matrix and address basic problem solved by computing the epipolar geometry.

3.3.1. The Essential Matrix

Given two calibrated cameras with calibration matrices K, K' and poses $[\mathrm{I} \mid 0]$ and $[\mathrm{R} \mid \mathrm{C}]$, and a 3d point M that is visible in both cameras as illustrated in fig. 3.5, we can relate the coordinates of M to its projection into both camera images by $\tilde{\mathrm{m}} = \mathrm{K}(\mathrm{I}\,0)\mathrm{M} = \mathrm{K}\bar{\mathrm{M}}$ and $\tilde{\mathrm{m}}' = \mathrm{K}'(\mathrm{R}^{\mathsf{T}} \; -\mathrm{R}^{\mathsf{T}}\mathrm{C})\mathrm{M}$. The normalized coordinates corresponding to the projection of the point into both images are given in the calibrated setting by projective points $\bar{\mathrm{m}}_{norm} = \mathrm{K}^{-1}\tilde{\mathrm{m}}$, and $\bar{\mathrm{m}}'_{norm} = \mathrm{K}^{-1}\tilde{\mathrm{m}}'$. The relationship between projections $\bar{\mathrm{m}}_{norm}$, $\bar{\mathrm{m}}'_{norm}$ and the corresponding 3d point M has to be explored to solve basic problems in stereo reconstruction:

- Estimating the position of $\bar{\mathrm{m}}'_{norm}$ given $\bar{\mathrm{m}}_{norm}$ and pose $[\mathrm{R} \mid \mathrm{C}]$ without knowing the actual position of the corresponding 3d point refers to *point matching* between camera images.

- Determinating the corresponding 3d point from given projections $\bar{\mathrm{m}}_{norm}$, $\bar{\mathrm{m}}'_{norm}$ and pose $[\mathrm{R} \mid \mathrm{C}]$ is referred to as *triangulation*.

- Reconstructing the inter-camera pose from $\bar{\mathrm{m}}_{norm}$, $\bar{\mathrm{m}}'_{norm}$ is performed by computing the Essential matrix which will be explored in the following.

As illustrated in fig. 3.5, the camera centers 0, C, projected points $\bar{\mathrm{m}}_{norm}$, $\bar{\mathrm{m}}'_{norm}$ and 3d point M are coplanar forming the so-called *epipolar plane*. The *epipole* denotes the point of intersection of the line joining the camera centers, referred to as the *baseline*, with the image plane. The epipole defines therefore the projection of the camera center

of each camera into each other's image plane. The intersection of the epipolar plane with each image plane is denoted as the *epipolar line*.

From the Euclidean relationship between the camera coordinate systems we obtain

$$\bar{\mathrm{m}}_{norm} = \mathrm{R}\bar{\mathrm{m}}'_{norm} + \mathrm{C}$$

With the cross product $\mathrm{C} \times \mathrm{R}\bar{\mathrm{m}}'_{norm}$ being perpendicular to C and $\mathrm{R}\bar{\mathrm{m}}'_{norm}$ we get

$$\bar{\mathrm{m}}(\mathrm{C} \times \mathrm{R}\bar{\mathrm{m}}') = 0$$

which accounts for the fact that the vectors between projections and the camera centers, $\overrightarrow{0, \bar{\mathrm{m}}_{norm}}$, $\overrightarrow{\mathrm{C}, \mathrm{R}\bar{\mathrm{m}}'_{norm} + \mathrm{C}}$, and the baseline vector $\overrightarrow{0, \mathrm{C}}$ are coplanar as shown in fig. 3.5. Using the skew-symmetric matrix representation $[\cdot]_\times$ of the cross product

$$\mathrm{C} \times \mathrm{C}' = \begin{pmatrix} \mathrm{C}_y\mathrm{C}'_z - \mathrm{C}_z\mathrm{C}'_y \\ \mathrm{C}_z\mathrm{C}'_x - \mathrm{C}_x\mathrm{C}'_z \\ \mathrm{C}_x\mathrm{C}'_y - \mathrm{C}_y\mathrm{C}'_x \end{pmatrix} = \begin{pmatrix} 0 & -\mathrm{C}_z & \mathrm{C}_y \\ \mathrm{C}_z & 0 & -\mathrm{C}_x \\ -\mathrm{C}_y & \mathrm{C}_x & 0 \end{pmatrix} \begin{pmatrix} \mathrm{C}'_x \\ \mathrm{C}'_y \\ \mathrm{C}'_z \end{pmatrix} = [\mathrm{C}]_\times \mathrm{C}' \quad (3.21)$$

this can be reformulated as:

$$\bar{\mathrm{m}}'^{\mathsf{T}}_{norm} \mathrm{E} \bar{\mathrm{m}}_{norm} = 0$$

where

$$\mathrm{E} = [\mathrm{C}]_\times \mathrm{R} \quad (3.22)$$

E defines the *Essential matrix* which is the algebraic representation of epipolar geometry for known camera calibration. It relates corresponding image points in different camera views with each other. The Essential matrix as defined by the constraint $\bar{\mathrm{m}}'^{\mathsf{T}}_{norm} \mathrm{E} \bar{\mathrm{m}}_{norm} = 0$ has only five degrees of freedom: Each translation C and rotation R has 3 degrees of freedom but there is an overall scale ambiguity since eq. (3.22) is defined on homogeneous coordinates.

3.3.2. Estimation of the Essential Matrix

Given two images of a camera with known intrinsic parameters, the Essential matrix is computed from corresponding normalized image points $\bar{\mathrm{m}}$, $\bar{\mathrm{m}}'$ such that the discrepancy from the epipolar constraint eq. (3.22) in terms of sum of squared distances between points and their corresponding epipolar lines is minimized.

To normalize corresponding pixel positions $\tilde{\mathrm{m}}$ in both images, the camera calibration matrix is applied: $\bar{\mathrm{m}}_{norm} = \mathrm{K}^{-1}\tilde{\mathrm{m}}$.

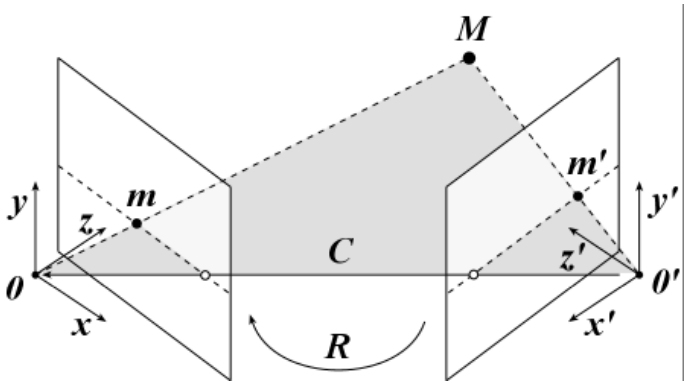
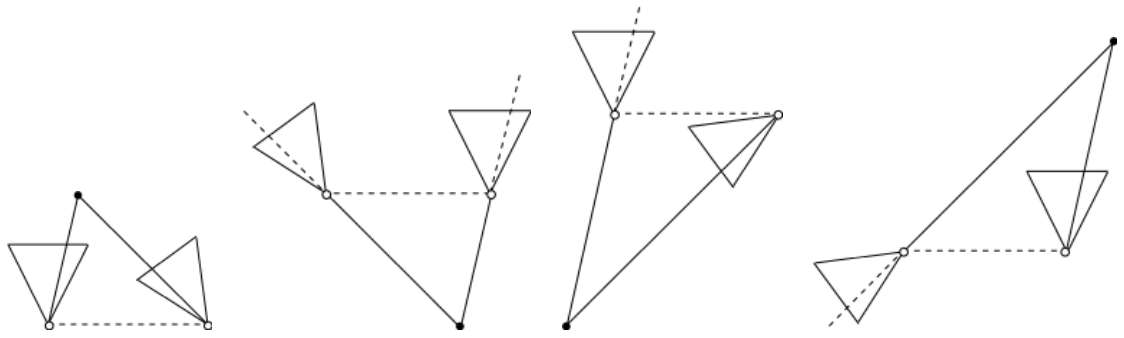


Figure 3.5.: Two cameras with epipolar plane

Once the Essential matrix E has been estimated, the normalized camera matrix $[\mathrm{R}|\mathrm{C}]$ containing the camera pose can be extracted from E up to scale and with a four-fold ambiguity where the different solutions represent the cases illustrated in fig. 3.6. This ambiguity is encountered as follows:

- The length of the position vector C of each solution is set to $\|\mathrm{C}\| = 1$ to define the indeterminate scale provided by the equation $\bar{\mathrm{m}}'^{\mathsf{T}}_{norm}\mathrm{E}\bar{\mathrm{m}}_{norm} = 0$.
- For each of the four solutions, the 3d points M corresponding to each 2d point pair $(\bar{\mathrm{m}}_{norm}, \bar{\mathrm{m}}'_{norm})$ are computed via triangulation as described in section 3.3.3.
- The solution with the most 3d points being *in front of both cameras* is selected (see fig. 3.6(a)).



(a) (b) (c) (d)

Figure 3.6.: The four possible solutions for pose reconstruction from the Essential matrix. Only in (a) the 3d point is actually visible in both camera.

For practical applications we use a RANSAC approach ([FB81], see also app. A.6) for robustness. We use the five-point algorithm proposed by *Nistér* [Nis03] to estimate an Essential matrix from 5 corresponding image point pairs. Image point correspondences

are established via the KLT feature tracking method proposed by *Kanade, Lucas and Tomasi* [TK91]. This issue will be explained in detail in section 3.5 regarding pose estimation.

3.3.3. Triangulation

Once the inter-camera pose is known, depth information can be recovered from point matches. Given a calibrated environment with inter-camera pose $[\mathrm{R} \mid \mathrm{C}]$, the 3d point $\bar{\mathrm{M}}$ corresponding to normalized projections $\bar{\mathrm{m}}_{norm}$, $\bar{\mathrm{m}}'_{norm}$ can be computed from the intersection of the rays through the camera centers and projected points[2] $\overrightarrow{0, \bar{\mathrm{m}}_{norm}}$ and $\overrightarrow{\mathrm{C}, \mathrm{R}\bar{\mathrm{m}}'_{norm} + \mathrm{C}}$ as shown below in fig. 3.7. Note that a scaling of the baseline lengths results in an equal isotropic scaling of the reconstructed 3d point space. Since the baseline length is set arbitrarily for poses recovered from Essential matrix estimation, 3d structure can only be reconstructed up to scale by this technique.

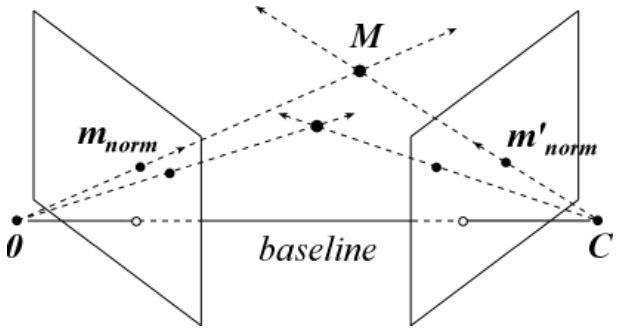


Figure 3.7.: Principle of depth measurement of image points by triangulation. The epipolar geometry has to be known to compute the 3d points from the intersection of the projection rays into the camera images.

3.4. Calibration of Intrinsic Parameters

In the last sections we assumed the intrinsic parameters of the camera to be known. The intrinsic parameters K can be estimated from corresponding 3d space and image entities using standard techniques as described in general in [HZ00]. Referring to the camera models described in section 3.2, the intrinsic parameters consist of focal length, principal point, skew, and lens distortion parameters for the perspective camera model,

[2] Since the rays from camera centers to $\bar{\mathrm{M}}$ and the baseline form a triangle, this technique for depth measurement is denoted as “triangulation”.

or principal point, radius scale, and distortion parameters for the spherical camera model.

In practice, the calibration for both camera models involves a planar checkerboard-type pattern with accurately known metrics which is referred to as *calibration grid* in the following. Given is a set of images $\mathcal{J}_1, \ldots, \mathcal{J}_K$ of the calibration grid captured from different unknown poses R^k, C^k of the camera ($k = 1, \ldots, K$) where the intrinsic parameters are assumed to be fixed over time. The corners of the grid are detected in the images by *Harris'* corner detector [HS88] and identified with the corresponding accurately known 3d points lying on the grid plane within the time-fixed coordinate frame of the grid $\mathcal{C}^{grid}$. $\mathcal{C}^{grid}$ is interpreted as the world coordinate frame for calibration.
From corresponding 3d points $\bar{\mathrm{M}}_j^k$ and 2d pixel positions $\tilde{\mathrm{m}}_j^k$ in each k-th image, the extrinsic and intrinsic parameters of the camera are estimated subject to minimizing the reprojection error - also denoted as *geometric error* - of 3d points into the images. The reprojection error is given by the squared error function:

$$\phi_{rp}^2(\mathrm{K}, \mathrm{R}^1, \mathrm{C}^1, \ldots, \mathrm{R}^K, \mathrm{C}^K) = \sum_{k=1}^{K} \sum_j \|\tilde{\mathrm{m}}_j^k - \mathrm{K}(\mathrm{R}^{k\mathsf{T}} \mid -\mathrm{R}^{k\mathsf{T}}\mathrm{C}^k)\bar{\mathrm{M}}_j^k\|^2 \qquad (3.23)$$

The estimation can in general be decomposed into the following steps:

1. First, find an initial solution to the intrinsic and extrinsic parameters. This is most commonly done by linear closed-form solutions computing the projection matrix from the linear equations provided by eq. (3.13) $\tilde{\mathrm{m}}_j^k \sim \mathbf{P}\bar{\mathrm{M}}_j^k$ for the perspective camera model. The projection matrix is decomposed into calibration matrix K and pose matrix $(\mathrm{R}^{k\mathsf{T}} \; -\mathrm{R}^{k\mathsf{T}}\mathrm{C}^k)$. Distortion is usually ignored to simplify this task.
   For the spherical camera model a linear approximation of the projection function $\mathcal{P}$ is used to determine an initial solution for the camera calibration function $\mathcal{K}$ from eq. (3.18).

2. Then the initial estimate is used as a starting point for minimizing the reprojection error with an iterative algorithm for non-linear optimization such as Levenberg-Marquardt.

The direct linear methods used for initialization is in general computionally fast because there are no iterations required. Nevertheless, its solution may not fulfill the parameter constraints in the presence of noise. Additional, lens distortion is modeled as a non-linear function and can generally only be approached by an iterative non-linear method.

For practical applications we use *Bouguet's* Camera Calibration Toolbox for Matlab® [Bou07] for intrinsic calibration of perspective cameras and stereo rigs, and an adaption of the Omnidirectional Camera Calibration Toolbox for Matlab® developed by *Scaramuzza et al.* [SMS06] to calibrate spherical cameras such as wide-angle fisheye-lens cameras. Both approaches are described in detail in the following sections.

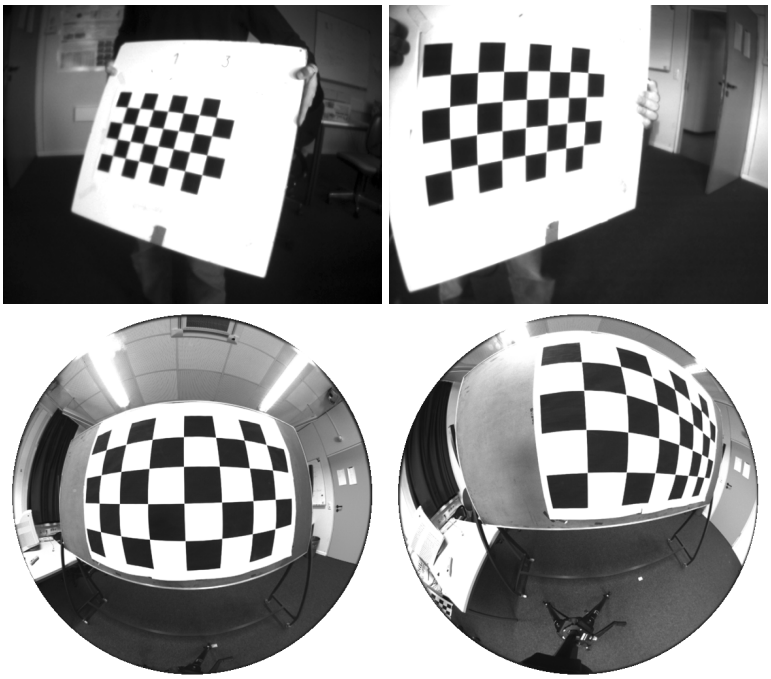


Figure 3.8.: Images for intrinsic camera calibration from corners of a calibration grid.
Top: Example views of the grid with a perspective camera for *Bouguet's* Camera Calibration toolbox with different camera poses.
Bottom: Example views of the grid with a fisheye-lens camera for the Omnidirectional Camera Calibration toolbox with different camera poses.

3.4.1. Calibration of Perspective Cameras

Calibration of intrinsic parameters with *Bouguet's* toolbox uses an intrinsic camera model equivalent to the one described in section 3.2.1 which approximates distortion as a fourth-order polynomial[3]. First, the extrinsic and intrinsic parameters apart from the distortion coefficients are estimated with a closed-form solution which examines

[3] *Bouguet* uses the "Plumb Bob" distortion model introduced by *Brown* which accounts for radial distortion and thin prism distortions, quod vide *D. C. Brown*: *"Decentering Distortion of Lenses"*, in: *"Photogrammetric Engineering, Vol. 32 (3)"*, p.444–462, 1966.

planar homographies between the images of the calibration grid as originally proposed by [Zha99].

The distortion function is estimated separately by minimizing a cost based on the deviation of edges on the calibration grid from fitting a linear model. The solution is afterwards refined iteratively by a non-linear optimization method regarding the distortion parameters subject to minimizing the geometric error as in eq. (3.23).

3.4.2. Calibration of Spherical Cameras

The Omnidirectional Camera Calibration toolbox for spherical cameras uses a similar camera model as described in 3.2.2. *Scaramuzza et al.* suggest to model distortion by describing the term $r \cot \Theta$ denoting the z-coordinate of the normalized term in eq. (3.20) by a fourth-order polynomial $z = \tau(r)$ instead of considering the viewing angle/pixel radius relationship $\Theta = \rho(r)$ as non-linear. First, the extrinsic parameters are estimated from $\mathrm{K}^{-1}(\tilde{\mathrm{m}}_j^k) \sim (\mathrm{R}^{k\mathsf{T}} \mid -\mathrm{R}^{k\mathsf{T}}\mathrm{C}^k)\bar{\mathrm{M}}_j^k$ arising from 2d-3d correspondences $\tilde{\mathrm{m}}_j^k, \bar{\mathrm{M}}_j^k$ for each image $k = 1, \ldots, K$. Using this constraint with eq. (3.20), the extrinsic parameters R^k, C^k for each image k are estimated from:

$$\begin{aligned} \mathrm{K}^{-1}(\tilde{\mathrm{m}}_j^k) \times (\mathrm{R}^{k\mathsf{T}} \mid -\mathrm{R}^{k\mathsf{T}}\mathrm{C}^k)\bar{\mathrm{M}}_j^k &= \begin{pmatrix} u_j^k - p_u \\ v_j^k - p_v \\ \tau(r) \end{pmatrix} \times (\mathrm{R}^{k\mathsf{T}} \mid -\mathrm{R}^{k\mathsf{T}}\mathrm{C}^k) \begin{pmatrix} X_j^k \\ Y_j^k \\ Z_j^k \\ 1 \end{pmatrix} = 0 \\ \Leftrightarrow \qquad \begin{pmatrix} 0 & -\tau(r) & v_j^k - p_v \\ & 0 & -u_j^k + p_u \\ -v_j^k + p_v & u_j^k - p_u & 0 \end{pmatrix} &(\mathrm{R}^{k\mathsf{T}} \mid -\mathrm{R}^{k\mathsf{T}}\mathrm{C}^k) \begin{pmatrix} X_j^k \\ Y_j^k \\ Z_j^k \\ 1 \end{pmatrix} = 0 \end{aligned} \tag{3.24}$$

where $r = \sqrt{(u_j^k - p_u) + (v_j^k - p_v)^2}$ and $\tau(r) = \kappa_0 + r\kappa_1 + \cdots + r^4\kappa_4$.

Note that eq. (3.24) yields one equation for each 2d-3d correspondence $\tilde{\mathrm{m}}_j^k, \bar{\mathrm{M}}_j^k$ that is linear in $\mathrm{R}^{k\mathsf{T}}$ and $\mathrm{t}^k := -\mathrm{R}^{k\mathsf{T}}\mathrm{C}^k$ but lacks row $\mathrm{R}_3^{k\mathsf{T}}$ and t_3^k.
First, the principal point is obtained by estimating the center of the view field circle. The extrinsic parameters are then estimated from the equations from eq. (3.24) being linear in $\mathrm{R}^{k\mathsf{T}}$ and t^k except for t_3^k (note that row $\mathrm{R}_3^{\mathsf{T}}$ can be computed from $\mathrm{R}_1^{\mathsf{T}}$ and $\mathrm{R}_2^{\mathsf{T}}$ since R^{T} is orthonormal). Afterwards the estimated extrinsic parameters are substituted in eq. (3.24) resulting in an overdetermined system of equations linear in

$\kappa_0, \ldots, \kappa_4$ and $\mathrm{t}_3^1, \ldots, \mathrm{t}_3^K$. This system of equations is solved as the previous by a linear least squares estimator utilizing singular value decomposition.
To maintain consistency with the distortion model proposed in section 3.2.2, the parameters of ρ can be estimated from τ by interpolating $\Theta_i = \frac{\pi}{2} - \arctan(\tau r_i, r_i)$ for several equidistant radii r_i.

For deeper insight into the issue of intrinsic camera calibration from planar calibration grids considering distortion models refer to the seminal work done by *Tsai* [Tsa87] and *Zhang* [Zha99]. See [SMS06] for a detailed description of the spherical camera calibration method used.

3.5. Pose Estimation and Structure from Motion

To perform the rig calibration as motivated it is essential to have time-corresponding poses of each camera in the rig each within its local reference frame. In this section we will describe how to obtain poses by the *structure from motion* approach for a single camera as proposed by e.g. [WHA93]. This approach is very generic and does not imply knowledge about the scene geometry or the presence of certain markers or calibration patterns.

Given is a camera with known intrinsic parameters K from a previously performed intrinsic camera calibration as described in section 3.4, and K images $\mathcal{J}_1, \ldots, \mathcal{J}_K$ captured during an arbitrary motion of the camera through an initially unknown scene. The task at hand is to estimate the poses $[\mathrm{R}^k \mid \mathrm{C}^k]$ of the camera for each time step $k = 1, \ldots, K$ with respect to the initial pose $[\mathrm{R}^0 \mid \mathrm{C}^0] = [\mathrm{I} \mid 0]$. All poses are described by Euclidean transformations of the camera-local coordinate frame with respect to the reference frame at time step 0. The task of pose estimation from images can be separated into the following subtasks:

Feature detection and tracking Interesting image points are extracted from the sequence of images and 2d-2d correspondences between subsequent images are computed. In order to simplify this task several image preprocessing steps are performed and motion between subsequent image is assumed to be small or motion preknowledge must be available.

Pose estimation initialization From 2d-2d correspondences an initial camera pose is computed by exploiting the Essential geometry, and sparse 3d structure is obtained up to scale. The absolute scale of the structure is defined arbitrarily.

Pose tracking By further tracing of corresponding image points, the following cam-

era poses for each image are computed from 2d-3d correspondences and the structure is updated consistently to the initially chosen scale.

3.5.1. Feature Detection and Tracking

In order to establish correspondences between image points in different images, a feature point detection and tracking approach is used.
Given an image $\mathcal{J}$, an image point $\tilde{\mathrm{m}} \in \mathcal{J}$ is considered as interesting for detection and matching if it can be distinguished sufficiently good from points in its vicinity. This applies to e.g. corners or fragments of texture. Such locally specific points are denominated as *feature points*. Next to the problem of determining certain points as feature points, referred to as *feature detection*, the issue of comparing and identifying images of the same feature point in different camera views arises in order to keep track of feature points over subsequential images from a moving camera. This is denoted as *feature matching* or *tracking*. There are different methods how to approach and identify feature points. Within the context of this work a KLT feature tracker is used for feature detection and matching as proposed and implemented by *Kanade, Lucas and Tomasi* [TK91] while *Harris'* corner detector [HS88] is used during intrinsic camera calibration to identify corners of the calibration grid. Briefly, the KLT method locates good feature points by examining the minimum eigenvalue of the structure tensor of image pixels within a small search window, and features are tracked by minimizing the difference between windows in subsequent images. Hence tracks of interesting image points over the image sequence are gained.
As stated in [TK91] the KLT feature tracker is theoretically robust against noise, image distortion, and change of lighting[4]. Feature points are located with sub-pixel accuracy resulting in smaller displacement errors. Also, tracking over Gaussian image pyramids with multiple resolution allows relatively large displacements between subsequent images even for small search windows.

3.5.2. Pose Estimation Initialization

Given image point correspondences, the pose estimation is initialized by estimating the Essential geometry between the first two images and setting up an initial sparse 3d structure. This will be referred to as *pose estimation initialization* (or in terms of the software used "init phase").

[4] In fact, in our practical applications we use a simple KLT implementation which is not robust against lighting changes since no image normalization is used. Nevertheless, it is appropriate for the rendered and real image sequences used for evaluation.

First, the Essential matrix $\mathrm{E}_{0,1}$ between the first two images $\mathcal{J}_0$, $\mathcal{J}_1$ is estimated from $n_{0,1}$ corresponding pairs of normalized 2d feature point positions extracted from the images $(\bar{\mathrm{m}}_1^0, \bar{\mathrm{m}}_1^1), \ldots, (\bar{\mathrm{m}}_{n_{0,1}}^0, \bar{\mathrm{m}}_{n_{0,1}}^1)$, using a RANSAC (vide app. A.6) for stability: Each sample solution $\mathrm{E}_{0,1}$ is computed via the five-point-algorithm proposed by *Nistér* [Nis03] from 5 corresponding point pairs as motivated in sec. 3.3.2. Correspondences $(\bar{\mathrm{m}}_i^0, \bar{\mathrm{m}}_i^1)$ are considered as outliers in the RANSAC framework if the residuum of $\bar{\mathrm{m}}_i^{1\mathsf{T}} \mathrm{E}_{0,1} \bar{\mathrm{m}}_i^0$ is larger than a fixed threshold for a sample solution $\mathrm{E}_{0,1}$.

The first camera defines the reference frame with initial pose $[\mathrm{I} \mid 0]$. The pose of the second camera with respect to the reference frame is computed from the estimated Essential matrix $\mathrm{E}_{0,1}$ as $[\mathrm{R}^1 \mid \mathrm{C}^1]$ with $\mathrm{E}_{0,1} = [\mathrm{C}^1]_\times \mathrm{R}^1$. As discussed in sec. 3.3.2, the Essential matrix can only be estimated up to scale hence the scale of the camera coordinate frame computed from $\mathrm{E}_{0,1}$ has an arbirary scale. This problem is addressed by scaling the first pose arbitrarily such that $\|\mathrm{C}^1\| = 1$, fixing the scale of the camera reference frame. From the poses $[\mathrm{R}^0 \mid \mathrm{C}^0]$ and $[\mathrm{R}^1 \mid \mathrm{C}^1]$ the 3d coordinates M_i corresponding to each feature point pair $(\bar{\mathrm{m}}_i^0, \bar{\mathrm{m}}_i^1)$ are estimated by triangulation (see section 3.3.3).

3.5.3. Pose Tracking

Given correspondences between image points and 3d points from the initialization phase, following poses are reconstructed maintaining consistency with the given 3d structure. This phase will be denoted as *pose tracking* (or in terms of the software used "tracking phase"). The notion behind the approach that is outlined below is to register the pose of the camera at each time step such that the reprojection error between 3d points and known corresponding image points becomes minimal.

After initializing we have not only the first two poses of the camera given in the camera reference frame but also correspondences between feature points and sparse 3d structure given within the reference frame. By further tracking of the feature points into the actual image, correspondences between normalizes image points $\bar{\mathrm{m}}_j^k$ and 3d points $\bar{\mathrm{M}}_j$ in the reference coordinate frame are obtained for each image $\mathcal{J}_k$. Recalling the projective relation $\bar{\mathrm{m}}_j^k \sim (\mathrm{R}^{k\mathsf{T}} \; - \mathrm{R}^{k\mathsf{T}} \mathrm{C}^k) \bar{\mathrm{M}}_j$ from eq. (3.13), the camera pose R^k, C^k of each image $\mathcal{J}_k$ with respect to the reference frame can be estimated by minimizing the reprojection error as defined in eq. (3.23), i.e. find $\mathrm{R}^k, \mathrm{C}^k$ such that:

$$\phi_{rp}^2(\mathrm{R}^k, \mathrm{C}^k) = \sum_{k=1}^{K} \sum_j \|\bar{\mathrm{m}}_j^k - (\mathrm{R}^{k\mathsf{T}} \mid - \mathrm{R}^{k\mathsf{T}} \mathrm{C}^k) \bar{\mathrm{M}}_j^k\|^2 \tag{3.25}$$

becomes minimal. Equation (3.25) is solved using a RANSAC for robustness. Each sample solution R^k, C^k is computed from 4 2d-3d correspondences by the POSIT

method proposed by *de Menthon and David* [MD95]. 2d-3d correspondences $\bar{\mathrm{m}}_i^k$, M_i are considered as outliers in the RANSAC framework if the reprojection error $\phi_{rp}^2(\mathrm{R}^k, \mathrm{C}^k)$ from eq. (3.25) is too large for a sample solution R^k, C^k.

To ensure that the algorithm does not lose track of the sparse 3d structure, new 3d points must be created during the pose tracking. For this purpose new feature points are detected and tracked as old feature points are lost. The 3d coordinates of new feature points are computed by triangulation as before. The algorithm tries to retrieve lost feature points in the actual image by predicting their position in $\mathcal{J}_k$ with the estimated pose.

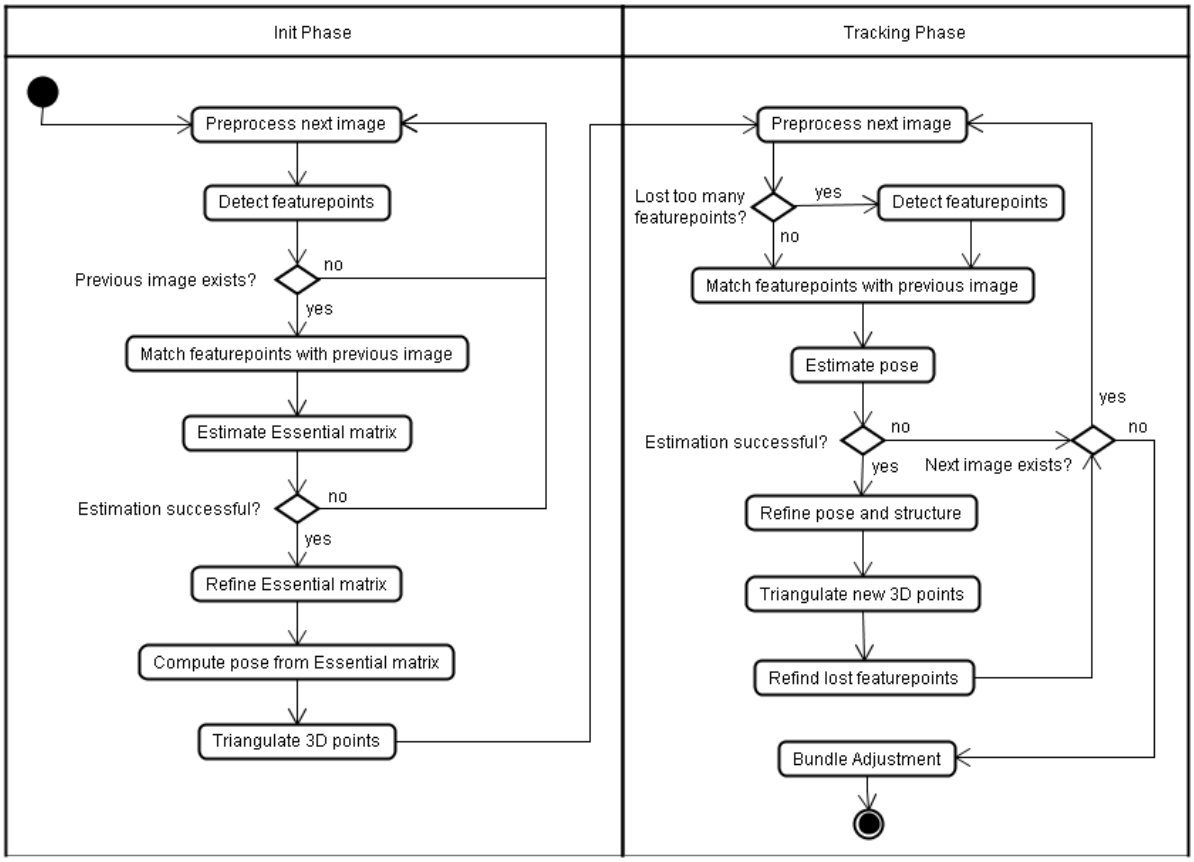


Figure 3.9.: Flowchart for pose estimation algorithm as described in section 3.5.
Init: 2d features from the initial image are tracked until the Essential matrix between the actual and the first image can be successfully estimated. The pose is computed from the Essential matrix, and 3d points are computed by triangulation.
Tracking: Poses are estimated from 2d-3d correspondences for each image and the 3d structure is extended by further tracked points.

The complete pose estimation algorithm is summarized in fig. 3.9. For robustness, all RANSAC estimations, i.e. Essential matrix estimation using the five-point-algorithm and pose estimation using the POSIT algorithm, are followed by a maximum-likelihood refinement. Also the reconstructed sparse 3d structure is updated during pose tracking by applying a Kalman filter. After the estimation has finished an additional bundle adjustment is performed using 2d-3d correspondences of all images to refine the computed poses and 3d structure. To explain these refinement techniques in detail would exceed the limitations of this work. For a detailed explanation of the refinement methods refer to [HZ00, WHA93]. A detailed description of the numerical methods for Essential matrix estimation and pose estimation can be found in [MD95, Nis03].

It is mentionable that during pose tracking the estimated poses and updated 3d structure are consistent with the scale fixed at initialization hence reconstructions of the same scene from different cameras differ by an overall scale factor modeled by different isometric scales of their respective reference coordinate systems. Nevertheless, although the pose error between subsequent frames is kept low, accumulating estimation errors can results in bias of the estimated poses and hence in a time-dependent error of the reconstruction scale. This problem will be covered in **Chapter 5** along with stability analysis of our approach.

3.6. Linear And Non-Linear Optimization

In order to approach real world problems arising from metrological applications in an analytical way, a mathematical model is developed which describes the relationship between measurements and the actual state of the observed system in terms of functional constraints between observation values and an appropriate parametrization of the system state.
In this chapter we will describe linear and non-linear models as well as analytical and numerical approaches to solve them for the system parameters given a certain set of observations.

3.6.1. Solving Systems of Linear Equations

A simple yet commonly used mathematical model is given by a system of linear equations (L) which is defined by a set of m linear equations in n real variables:

$$\mathrm{Ax} = \mathrm{b}$$

or

$$\mathrm{A}_j \mathrm{x} = b_j \quad \text{for each } j = 1, \ldots, k \tag{3.26}$$

where A is a real $m \times n$ matrix consisting of m row vectors $\mathrm{A}_1^\mathsf{T}, \cdots, \mathrm{A}_m^\mathsf{T} \in \mathbb{R}^n$, $\mathrm{b} \in \mathbb{R}^m$ is a column vector of observations and $\mathrm{x} \in \mathbb{R}^n$ is a column vector of unknowns. x accounts for the a priori unknown system parameters in general.

While accurate equation systems with full rank can be solved easily, metrological applications have to deal with noise-afflicted and erroneous data. Hence it is mandatory to use analytical or numerical optimization methods to compensate for the input errors and obtain a valid solution.

Error model for systems of linear equations: Given a linear equation $\mathrm{Ax} = \mathrm{b}$, describing the model behavior, and a measurement $\hat{\mathrm{b}}$ afflicted with unknown measurement errors, a common error model is

$$\hat{\mathrm{b}} = \mathrm{b} + \mathrm{e} \tag{3.27}$$

where e accounts for the error between measured and "real" observations which is in general unknown.
We are looking for the parameter vector $\hat{\mathrm{x}}$ satisfying $\mathrm{A}\hat{\mathrm{x}} + \hat{\mathrm{e}} = \hat{\mathrm{b}}$ that minimizes the residual vector $\|\hat{\mathrm{e}}\|$ [5].

3.6.2. Solving Linear Least Squares Problems

In most cases we have more observations than constraints on system parameters. Observations are afflicted with an unknown error resulting from measurement inaccuracies, numeric limitations or previous estimation errors. These errors are thought of as *noise* on the input data for solution algorithms. Consider an over-determined noisy system of linear equations (L) with n unknowns $\mathrm{x} \in \mathbb{R}^n$ and $m \geq n$ linear equations $\mathrm{A}_j \mathrm{x} = b_j$ with $\mathrm{A}_j^\mathsf{T} \in \mathbb{R}^n$, $b_j \in \mathbb{R}$ for each $j = 1, \ldots, m$.

Assuming that the noise is Gaussian distributed with zero mean, the most likely solution for the equation system with respect to the linear error model stated in eq. (3.27)

[5] Notice the difference between residuals and measurement error: $\hat{\mathrm{e}}$ describes the error between observations expected from the *estimated* parameters and measured observations while e describes the error between observations expected from the *real* parameters and measured observations.

can be found by solving the *linear least squares problem* imposed by (L) of minimizing the residuals $|\mathrm{A}_j\mathrm{x} - b_j|$ for each $j = 1, \ldots, m$, i.e. to minimize the squared error function:

$$\phi^2(\mathrm{x}) = \|\mathrm{Ax} - \mathrm{b}\|^2 = \sum_{j=1}^{m} (\mathrm{A}_j\mathrm{x} - b_j)^2 \tag{3.28}$$

To solve the linear least squares problem one takes advantage of the fact that ϕ^2 reaches its minimum at x^* when its partial derivatives with respect to the parameters $x_1, \ldots, x_n$ reach zero:

$$\frac{\partial}{\partial \mathrm{x}} \phi^2(\mathrm{x}^*) = 0 \tag{3.29}$$

From this constraint one obtains the *normal equations* of (L) which provide the solution x^* for (L):

$$\begin{array}{rrcl} & \frac{\partial}{\partial x}\phi^2(\mathrm{x}^*) & = & 2\mathrm{A}^\mathsf{T}\mathrm{Ax}^* - 2\mathrm{A}^\mathsf{T}\mathrm{b} = 0 \\ \Leftrightarrow & \mathrm{A}^\mathsf{T}\mathrm{Ax}^* & = & \mathrm{A}^\mathsf{T}\mathrm{b} \end{array} \tag{3.30}$$

If A has full column rank, the square matrix $\mathrm{A}^\mathsf{T}\mathrm{A}$ can be inverted and x^* is obtained by applying the *pseudoinverse* $\mathrm{A}^\dagger = (\mathrm{A}^\mathsf{T}\mathrm{A})^{-1}\mathrm{A}^\mathsf{T}$ of A to b:

$$\mathrm{x}^* = (\mathrm{A}^\mathsf{T}\mathrm{A})^{-1}\mathrm{A}^\mathsf{T}\mathrm{b} \tag{3.31}$$

There are several efficient numerical approaches to solve linear least squares problems such as by QR or LQ factorization, by complete orthogonal decomposition or by singular value decomposition [WR71]. We use the *singular value decomposition* (SVD) because in the presence of noise we do not know the rank of A and cannot assure that A is not rank-deficient. The SVD offers a high numerical accuracy and stability even for ill-conditioned matrices and can find solutions when other methods fail. It is, however, computationally expensive. Another criterion for using the SVD for solving linear least squares problems is that is dispenses with inverting the symmetric form $\mathrm{A}^\mathsf{T}\mathrm{A}$ explicitly which avoids one source of numerical instability.
Briefly, the SVD computes a factorization of a rectangular matrix A that allows to express inter alia the pseudoinverse $\mathrm{A}^\dagger$ by which the linear least squares problem can be

solved using eq. (3.31). For a detailed description of the SVD and applications to different problems such as solving systems of linear equations, eigenvalue determination, and computation of the pseudoinverse refer to appendix A.1 or [WR71].

Error propagation

Consider a linear least squares problem corresponding to $\mathrm{A}\mathrm{x} = \mathrm{b}$ with squared error function $\phi^2(\mathrm{x}) = \|\mathrm{A}\mathrm{x} - \hat{\mathrm{b}}\|^2$ for $\mathrm{A} \in \mathbb{R}^{m \times n}$, $\hat{\mathrm{b}} \in \mathbb{R}^m$ and unknowns $\mathrm{x} \in \mathbb{R}^n$. As stated above the linear least square estimator yields $\hat{\mathrm{x}} = \mathrm{A}^\dagger \hat{\mathrm{b}}$, involving the pseudoinverse $\mathrm{A}^\dagger = (\mathrm{A}^\mathsf{T}\mathrm{A})^{-1}\mathrm{A}^\mathsf{T}$. Using the error model for linear equations defined in eq. (3.27) we consider that $\hat{\mathrm{b}}$ is afflicted with an error e. Relating the linear least squares estimate with the "real" solution x we have $\hat{\mathrm{x}} = \mathrm{A}^\dagger \hat{\mathrm{b}} = \mathrm{A}^\dagger(\mathrm{b} + \mathrm{e}) = \mathrm{x} + \mathrm{A}^\dagger \mathrm{e}$, hence the error propagated from the measurements $\hat{\mathrm{b}}$ to the parameter solution $\hat{\mathrm{x}}$ is linearly transformed by $\mathrm{A}^\dagger$.
Assuming the measurement errors e to be Gaussian distributed with zero mean, $e_j \sim \mathcal{N}(0, \sigma_j)$ for each $j = 1, \ldots, m$, the error ε propagated to $\hat{\mathrm{x}}$ is also Gaussian distributed with $\varepsilon_i \sim \mathcal{N}(0, \mathrm{A}_i^\dagger \sigma)$ for each $i = 1, \ldots, n$ where $\mathrm{A}_i^\dagger$ denotes the i-th row of $\mathrm{A}^\dagger$.

Assuming that A is also depending on measurements and hence afflicted with measurement errors, the error model for linear equations defined in eq. (3.27) cannot be applied. The model can be augmented by denoting the erroneous matrix by $\hat{\mathrm{A}}$ and the measurement error of each component $\hat{\mathrm{A}}_{i,j}$ by the matrix E while A refers to the "real" matrix. The linear least squares estimator yields $\hat{\mathrm{x}} = \hat{\mathrm{A}}^\dagger \hat{\mathrm{b}} = \hat{\mathrm{A}}^\dagger(\mathrm{b} + \mathrm{e}) = \mathrm{x} + \hat{\mathrm{A}}^\dagger \mathrm{e}$, hence the propagated error is $(\mathrm{AE})^\dagger \mathrm{e}$. Error propagation for such cases is in general intractable to analyze analytically and is usually replaced by numeric evaluation techniques such as computing the output error for a large number of input instances afflicted with error. Such techniques are commonly referred to as *Monte Carlo techniques*.

3.6.3. Solving Non-Linear Least Squares Problems

For linear least squares problems the components of the error vector are linear functions of the unknowns x hence linear algebra tools can be used to find a solution. Now consider that the error vector consists of non-linear functions of x. The general form of a non-linear least squares error function is given by

$$f(\mathrm{x}) = \|\mathrm{g}(\mathrm{x})\|^2 = \sum_{j=1}^{m} g_j(\mathrm{x})^2 \tag{3.32}$$

where $\mathrm{g} : \mathbb{R}^n \to \mathbb{R}^m, \mathrm{x} \mapsto (g_j(\mathrm{x}))_{j=1,\ldots,m}$ defines a non-linear vector function and each $g_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \ldots, m$ is a non-linear function over the unknowns x.

As the linear least squares method can be used to estimate solutions of disturbed linear equation systems, non-linear equation systems $\mathrm{y} = \mathrm{h}(\mathrm{x})$ can be modeled as a non-linear least squares problem by minimizing the error function $f(\mathrm{x}) = \|\mathrm{g}(\mathrm{x})\|^2$ with $\mathrm{g}(\mathrm{x}) = \mathrm{h}(\mathrm{x}) - \mathrm{y}$. To enforce non-linear implicit constraints $h_j(\mathrm{x}, \mathrm{y}) = 0$ between system parameters x and observations y one may solve the non-linear least squares problem with error function components $g_j(\mathrm{x}) = h_j(\mathrm{x}, \mathrm{y})$.

There are several optimization methods designed specially for solving non-linear least squares problems. For a detailed overview of numerical approaches associated with this problem refer to [MNT04]. The most commonly used methods for solving non-linear least squares problems are *Newton's approach* and its variations, e.g. the *Levenberg-Marquardt method*. An implementation of the Levenberg-Marquardt method as described in [PFTW88] will be used in this work for practical applications.

Newton's approach: Newton's approach starts with an initial parameter solution x_0 and refines this vector iteratively based on the assumption that g is locally linear by evaluating a first order approximation g at x_0 with a small step $\Delta\mathrm{x}$: $\mathrm{g}(\mathrm{x}_0 + \Delta\mathrm{x}) \approx \mathrm{g}(\mathrm{x}_0) + \nabla\mathrm{g}(\mathrm{x}_0)\Delta\mathrm{x}$ where $\nabla\mathrm{g}(\mathrm{x}_0)$ is the Jacobian of g evaluated at x_0. Hopefully, the algorithm converges to the desired globally minimal solution but it is also possible that it ends up in a local minimum or does not converge at all.

Levenberg-Marquardt method: The Levenberg-Marquardt method is basically a damped version of Newton's approach. While Newton's method uses fixed steps $\Delta\mathrm{x}$, in each Levenberg-Marquardt iteration step $\Delta\mathrm{x}$ is adapted according to its influence on the error. However, the result depends also largely on the initial solution x_0. Such an approach is in general only efficiently appliable if x_0 is sufficiently close to an optimal solution.

4. Multi-Camera Rig Calibration

In this main chapter we explore the task of *rig calibration* of a fixed multi-camera rig where "fixed" denotes that their internal parameters and relative positions of each camera do not change over time.

There are several calibration approaches present for stereo-rigs with overlapping field of view. Most approaches separate this problem into calibration of the intrinsic parameters of each camera and determination of the relative poses of the camera inside the rig, the latter denoted here as *rig parameters*. The first issue can be most efficiently solved using calibration patterns with known metrics ([Tsa87, Bou07, SMS06]). For the second task different approaches are conceivable, ranging from using calibration objects to automatic calibration from a sequence of time-corresponding images of all cameras during an unknown movement of the rig through an arbitrary setting.

This problem is originally based on applications of stereography. However, in general one assumes that the coupled cameras have overlapping view fields. Under this condition, points that have been visible in multiple cameras can be used to apply stereographic methods in order to obtain the relative poses of the cameras pairwise, e.g. from the epipolar constraints between the cameras.

In contrast to this, for the proposed problem it should *not* be assumed that the coupled cameras have overlapping fields of view. Hence we cannot utilize methods that rely on points seen by multiple cameras from the beginning. On the contrary, we will utilize results from the field of *hand-eye calibration*: We use constraints between the local poses of each slave camera in the rig and the local poses of the master camera during a movement of the rig to estimate the rig parameters without assuming overlapping views. Local poses will be delivered by image-based pose estimation. Once a sufficient good estimate for the rig parameters has been found structure of the scene can be recovered from the combined views of all cameras as proposed in [FKK04] or [Boe07].

First we will define a model of the problem of multi-camera rig calibration and introduce the notations used. Then we will design a framework for an algorithm to solve this task under certain assumptions.

4.1. Definition of the Multi-Camera Rig

A so called (fixed) *multi-camera rig* is defined by a rigid coupling of multiple cameras such that motions of the whole rig leave the poses of the rig cameras relative to each other unchanged. Figure 4.1 and fig. 4.2 show configurations of multi-camera rigs that are used in practical applications. In this section we will give a precise description of the rig model that will be used in this work.

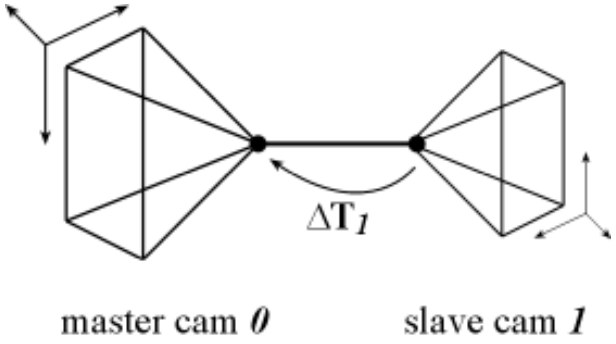


Figure 4.1.: Stereo-camera rig consisting of two perspective cameras with non-overlapping views.

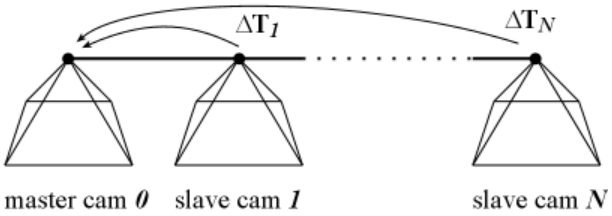


Figure 4.2.: Typical multi-camera rig for 3d scene reconstruction consisting of $N+1$ perspective cameras with non-overlapping views.

Given are $N+1$ cameras modeled e.g. by the perspective or spherical camera model as described in section 3.2. The cameras describe a rigidly coupled multi-camera-system called a *multi-camera rig* if their poses are fixed inside the rig's internal coordinate frame $\mathcal{C}^{rig}$. In other words, the position and orientation of the coordinate frames $\mathcal{C}^i$ of each camera i with respect to $\mathcal{C}^{rig}$ is constant over time. Although the origin and orientation of the rig's coordinate frame $\mathcal{C}^{rig}$ could be chosen arbitrarily, we will, for convenience, designate the camera with index 0 as *master camera* which defines the rig's coordinate frame, while the other cameras with indices $1, \ldots, N$ will be referred to as *slave cameras*. Their fixed poses inside the rig with respect to the master camera's pose will be referred to as *internal poses* in the following.

Precisely, the fixed relative pose of each slave camera $i = 1, \ldots, N$ with respect to master camera 0 is given by its *internal position* $\Delta\mathrm{C}_i = (\Delta x_i, \Delta y_i, \Delta z_i)^\mathsf{T}$ and its *internal orientation* $\Delta\mathrm{R}_i$ within the rig coordinate frame $\mathcal{C}^{rig}$. Furthermore, each slave

camera is associated with an *internal scale* factor $\Delta\lambda_i$ which relates the size of its coordinate frame $\mathcal{C}^i$ to the rig coordinate frame $\mathcal{C}^{rig}$. This parameter is mandatory since the corresponding poses from which the system will be calibrated do not necessarily have to be measured in coordinate frames of equal scale. Obviously, because we identify the rig coordinate frame with the master camera coordinate frame, the master camera has internal position $\Delta\mathrm{C}_0 = 0$, internal orientation $\Delta\mathrm{R}_0 = \mathrm{I}$ and internal scale $\Delta\lambda_0 = 1$. Optionally, a scale relating the rig coordinate frame with the metrics of a wrapping world coordinate frame can be defined by $\Delta\lambda_{world}$, given e.g. in millimeters per rig coordinate frame unit.

The rotation parameters may be expressed in various ways, e.g. by rotation axis $\Delta\mathrm{r}_i$ and rotation angle $\Delta\alpha_i$, by its Euler angles $\Delta\alpha_i$, $\Delta\beta_i$, $\Delta\gamma_i$, or by a unit quaternion $\Delta\mathbf{q}_i$ [FW04]. The best representation for rotation will be discussed later in section 4.3.1. A more detailed discussion of this topic is presented in appendix A.2.

Internal positions, orientations, and scales for each slave camera will be referred to as the *rig parameters* in the following.

As introduced in 3.1.2 the similarity transformation between the local coordinate frames of slave camera i and master camera 0 is given by the 4×4 matrix:

$$\Delta\mathbf{T}_i = \begin{pmatrix} \Delta\lambda_i\Delta\mathrm{R}_i & \Delta\mathrm{C}_i \\ 0^\mathsf{T} & 1 \end{pmatrix}$$

or for short $\Delta\mathbf{T}_i = [\Delta\lambda_i\Delta\mathrm{R}_i \mid \Delta\mathrm{C}_i]$.

The inverse transformation is given by the 4×4 matrix

$$\Delta\mathbf{T}_i^{-1} = \begin{pmatrix} \frac{1}{\Delta\lambda_i}\Delta\mathrm{R}_i^\mathsf{T} & -\Delta\mathrm{R}_i^\mathsf{T}\Delta\mathrm{C}_i \\ 0^\mathsf{T} & 1 \end{pmatrix}$$

or for short $\Delta\mathbf{T}_i^{-1} = [\frac{1}{\Delta\lambda_i}\Delta\mathrm{R}_i^\mathsf{T} \mid -\Delta\mathrm{R}_i^\mathsf{T}\Delta\mathrm{C}_i]$.

Connection to hand-eye estimation: The subject of hand-eye estimation is basically given by a rigid coupling of devices, commonly a camera (“eye”) and a kind of gripper (“hand”) as illustrated in **Section 2**, fig. 2.1. Such devices are described in an equivalent way as a multi-camera rig by the internal poses of hand and eye with respect to each other. Since the rig model proposed above is a straightforward generalization of the hand-eye model, it will be used for both throughout this work. However,

most hand-eye calibration approaches imply that all measurements are taken within the same metric framework.

It is an essential property of the proposed rig model that relations between coordinate frame of cameras are considered by similarity transformations rather than by Euclidean transformations. Thus, the model is generic and is not restricted to the condition that observations in each camera frame are measured with respect to the same metric scale. Later, we will discuss a further generalization of the rig model allowing for the scale of the camera reference frames to change over time.

In general, estimation of intrinsic parameters and rig parameters is done separately. First, intrinsic camera calibration depending on the appropriate camera model is performed for each camera $i = 0, \ldots, N$ in the rig individually as described in section 3.4. Note that once the camera calibration functions K_i are known, we will not depend on the specific camera model of the rig cameras for the rest of this chapter.
The second calibration task consists of computing the rig parameters $\Delta\mathrm{R}_i$, $\Delta\mathrm{C}_i$, and $\Delta\lambda_i$. In the next section we will explore constraints between corresponding poses of the coupled cameras in order to address this issue.

4.2. Pose Constraints of the Moving Rig

In this section we will describe the relationship between the rig parameters, i.e. the internal poses of each camera in the rig with respect to the master camera, and pose transformations in the camera-local coordinate frames observed during a motion of the rig in a similar way as for the hand-eye calibration problem.

We consider that the rig is moving over time, precisely: For each time step $k = 0, \ldots, K$ we have different external poses R_{rig}^k, C_{rig}^k within in the world coordinate frame $\mathcal{C}^{world}$. First we will explore the relationship between camera poses and rig parameters considering different settings for the individual camera poses. We will denote these as *pose models*:

Global reference poses Assume that the poses of *all* cameras $0, \ldots, N$ at each time step $k = 0, \ldots, K$ are given with respect to the world coordinate frame as reference frame as illustrated in fig. 4.3. The pose for camera i at time step k is defined by a Euclidean transformation $\mathbf{T}_i^k = [\mathrm{R}_i^k \mid \mathrm{C}_i^k]$ with respect to $\mathcal{C}^{world}$. Such poses are received e.g. from pose estimation with a reference object visible in all cameras at the same time which defines the world coordinate frame. Then the relationship between master and slave camera poses and rig parameters are

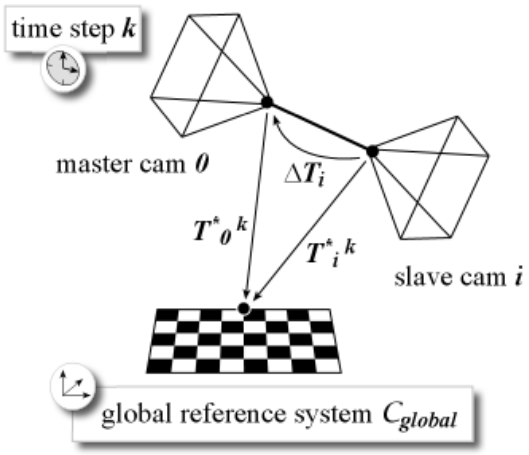


Figure 4.3.: Rig parameters vs. camera poses in global reference frame

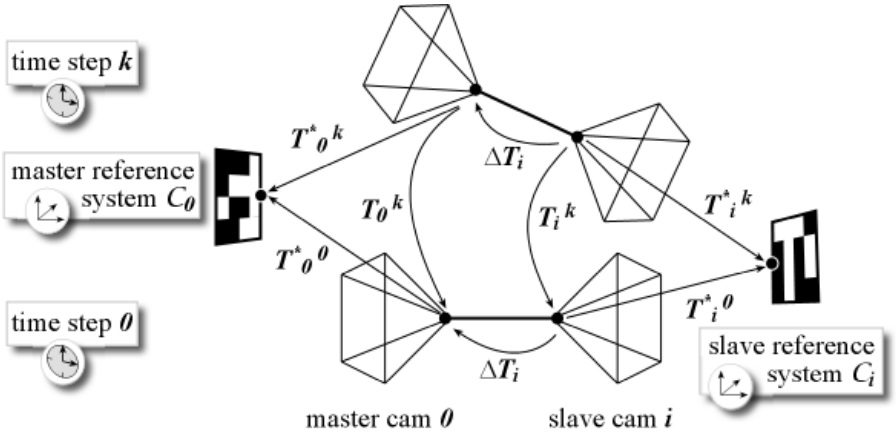


Figure 4.4.: Rig parameters vs. camera poses in individual reference frames

given by a Euclidean transformation $\Delta\mathbf{T}_i$ for which holds $\mathbf{T}_0^k \Delta\mathbf{T}_i = \mathbf{T}_i^k$, or more explicitly:

$$[\mathrm{R}_0^k \mid \mathrm{C}_0^k][\Delta\mathrm{R}_i \mid \Delta\mathrm{C}_i] = [\mathrm{R}_i^k \mid \mathrm{C}_i^k]$$

This pose model yields a very simple constraint on the rig parameters and allows to compute them by solving the linear equation $\Delta\mathbf{T}_i = (\mathbf{T}_0^k)^{-1}\mathbf{T}_i^k$ considering Euclidean transformations only. This method can be used for stereo rig calibration e.g. with *Bouguet's* toolbox [Bou07] from images with large overlapping views. Nevertheless, this pose model is not feasible for the assumption of non-overlapping views because it is very difficult to apply a reference object with accurately known metrics such that it is visible in all cameras at the same time in general.

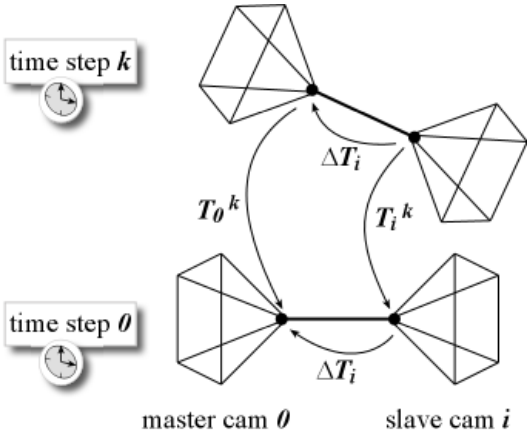


Figure 4.5.: Rig parameters vs. camera poses in camera-local reference frames

Individual reference poses Assume that the pose of each camera $i = 0, \ldots, N$ at each time step $k = 0, \ldots, K$ is given by a Euclidean transformation $\mathbf{T}_i^k$ with respect to a *camera-specific* reference frame $\mathcal{C}^i$ as illustrated in fig. 4.4. Such poses are received e.g. from pose estimation where for each camera in the rig an individual reference object is visible at all time steps which defines the respective camera reference frame. The relationship between master and slave camera poses and rig parameters is then given by a Euclidean transformation $\Delta\mathbf{T}_i$ for which holds $(\mathbf{T}_0^0)^{-1}\mathbf{T}_0^k\Delta\mathbf{T}_i = \Delta\mathbf{T}_i(\mathbf{T}_i^0)^{-1}\mathbf{T}_i^k$, or more explicitly:

$$[\mathrm{R}_0^{0\mathsf{T}} \mid -\mathrm{R}_0^{0\mathsf{T}}\mathrm{C}_0^0][\mathrm{R}_0^k \mid \mathrm{C}_0^k][\Delta\mathrm{R}_i \mid \Delta\mathrm{C}_i] = [\Delta\mathrm{R}_i \mid \Delta\mathrm{C}_i][\mathrm{R}_i^{0\mathsf{T}} \mid -\mathrm{R}_i^{0\mathsf{T}}\mathrm{C}_i^0][\mathrm{R}_i^k \mid \mathrm{C}_i^k]$$

for each slave camera i at time step k.

Although this approach can be applied to multi-camera rigs without overlapping fields of view it is difficult to realize since the rig motion is very limited due to the fact that the respective calibration objects have to be visible in all images. We will refer to this method later to evaluate our approach for non-overlapping views.

Camera-local poses The most appropriate pose model for our approach is the one commonly used for hand-eye estimation as depicted in fig. 4.5: The reference frame for each camera i is defined by the initial pose of the camera and following poses are given with respect to the initial camera pose. Such poses are received e.g. from pose estimation without certain reference objects as described in sec. 3.5 and resemble the local poses referred to in hand-eye calibration approaches [1]. As motivated above, the rig coordinate frame is identified with the reference frame of the master camera, $\mathcal{C}^{rig} = \mathcal{C}^0$.

[1] In hand-eye calibration, the local poses of the "hand" are commonly obtained from an observing camera while local poses of the "eye" camera are computed by image processing methods equal to

As stated above, we identify the reference frame for each camera with its initial coordinate frame and relate local pose transformation of the cameras with each other for simplicity, as commonly done in hand-eye calibration approaches. The following considerations are meant to be generic and do not depend on the concrete technique of pose acquisition.

The reference frame for each camera inside the rig shall be defined throughout this work by its initial position at time step $k = 0$ scaled isometrically by an arbitrary scale factor. The reference frame for each camera $i = 0, \ldots, N$ is denoted as $\mathcal{C}^i$. For each time step $k = 0, \ldots, K$ the pose of camera i with respect to its reference frame is denoted as R_i^k, C_i^k. By the definition of the reference frame we have $\mathrm{R}_i^0 = \mathrm{I}$ and $\mathrm{C}_i^k = 0$ for each camera i.

Considering that each moving camera changes its pose over time but not the scale of its camera-local coordinate frame with respect to its reference frame, the relation between the coordinate frame of each camera $i = 0, \ldots, N$ at time step $k = 1, \ldots, K$ and its reference frame is given by the Euclidean transformation

$$\mathbf{T}_i^k = \begin{pmatrix} \mathrm{R}_i^k & \mathrm{C}_i^k \\ 0^\mathsf{T} & 1 \end{pmatrix}$$

or, in short, by $\mathbf{T}_i^k = [\mathrm{R}_i^k \mid \mathrm{C}_i^k]$.

In section 4.1 we identified the similarity transformation $\Delta\mathbf{T}_i$ relating the coordinate frames of slave camera i and master camera 0. Because this relation is fixed over time, we can relate the coordinate frames of cameras at each point of time via $\Delta\mathbf{T}_i$ as visible in fig. 4.5.

Hence, we get for each time step $k = 1, \ldots, K$ and each slave camera $i = 1, \ldots, N$ the following equality constraint:

$$\boxed{\mathbf{T}_0^k \Delta\mathbf{T}_i = \Delta\mathbf{T}_i \mathbf{T}_i^k}$$

which can be expressed as:

$$\begin{pmatrix} \mathrm{R}_i^0 & \mathrm{C}_i^0 \\ 0^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \Delta\lambda_i \Delta\mathrm{R}_i^k & \mathrm{C}_i^k \\ 0^\mathsf{T} & 1 \end{pmatrix} = \begin{pmatrix} \Delta\lambda_i \Delta\mathrm{R}_i^k & \mathrm{C}_i^k \\ 0^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \mathrm{R}_i^k & \mathrm{C}_i^k \\ 0^\mathsf{T} & 1 \end{pmatrix} \tag{4.1}$$

the one used for our approach. Hence, a multi-camera rig is colloquially interpreted as a hand-eye device consisting of “eyes” only.

From eq. (4.1) we get the following constraints for the rig parameters:

$$\mathrm{R}_0^k \Delta\mathrm{R}_i = \Delta\mathrm{R}_i \mathrm{R}_i^k \tag{4.2}$$

$$\mathrm{R}_0^k \Delta\mathrm{C}_i + \mathrm{C}_0^k = \Delta\lambda_i \Delta\mathrm{R}_i \mathrm{C}_i^k + \Delta\mathrm{C}_i \tag{4.3}$$

Apart from the scales $\Delta\lambda_i$, this is identical to the equations that have to be solved in hand-eye calibration approaches as in [HD95].

4.2.1. Motion Models

Throughout this work we will distinguish applications where the rig motion is purely translational without notable rotation from applications where general motion containing rotation and translation is performed. As it will be pointed out, the rig calibration problem has to be treated differently depending on the type of motion. We define two different motion models as follows:

General Motion Model Assuming general motion we have $\mathrm{R}_i^k \neq \mathrm{I}$ at each time step $k = 1, \ldots, K$ for each camera $i = 0, \ldots, N$.

Purely Translational Motion Model A purely translating rig is modeled by assuming $\mathrm{R}_i^k = \mathrm{I}$ for each time step $k = 1, \ldots, K$ and camera $i = 0, \ldots, N$.

We develop different calibration approaches for both models. For practical applications the appropriate model has to be chosen. In section 5.2 the performance of both motion models will be evaluated under different amounts of rig rotation, and a method for model selection will be given.

4.2.2. Scale Models

As mentioned in section 3.5, due to error accumulation during pose estimation the reconstructed poses are afflicted with a time-dependent scale error. In the general rig parameters model we assume the scale $\Delta\lambda_i$ between the reference coordinate frame of each slave camera i and the master camera reference coordinate frame to be constant. Another setting where time-dependent scales occur is given when the pose estimation method is reinitialized at certain steps of time, e.g. if the pose estimation lost track of

all 3d points. To consider strong scale deviations over time or redefinition of the scale at certain time steps we will distinguish two different scale models:

Time-Fixed Scale Model The general scale model assumes the scale $\Delta\lambda_i$ to be constant over time. The transformation of the camera frame of slave camera i at time step k to the master camera frame is given by a time-independent similarity transformation $[\Delta\lambda_i \Delta\mathrm{R}_i \mid \Delta\mathrm{C}_i]$.

Time-Dependent Scale Model To model conditions where the scale is known to change significantly over time we assume the scale to be locally constant during n intervals of time. We define a *time set* given by a partition of $\{1, \ldots, K\}$ into subsets $\mathcal{K}^{(1)}, \ldots, \mathcal{K}^{(n)}$. At each time step $k \in \mathcal{K}^{(\nu)}$, $\nu = 1, \ldots, n$, the transformation of the camera frame of slave camera i to the master camera frame is given by a similarity transformation $[\Delta\lambda_i^{(\nu)} \Delta\mathrm{R}_i \mid \Delta\mathrm{C}_i]$. Hence the number of rig parameters is increased by $n - 1$ with respect to the time-fixed scale model. The time-fixed model can be interpreted as a special case of the time-dependent scale model with $n = 1$ and $\mathcal{K}^{(1)} = \{1, \ldots, K\}$. The worst case scenario is given by $n = K$ and $\mathcal{K}^{(\nu)} = \{\nu\}$ for each $\nu = 1, \ldots, n$ where the internal scale is considered to be arbitrary for any frame k.

For any given practical application the appropriate model has to be chosen. In section 5.3 the performance of time-fixed and time-dependent scale model will be evaluated under different scale deviations over time, and a method for model selection will be given.

4.3. Calibration of Rig Parameters

The general rig calibration algorithm proceeds as shown in fig. 4.6. First the intrinsic parameters of each camera are calibrated individually by common techniques as described in section 3.4. The remaining task of *rig calibration* is now defined by computing the parameters of the rigid coupling, i.e. the relative poses $\Delta\mathrm{C}_i$, $\Delta\mathrm{R}_i$, and scales $\Delta\lambda_i$ of each slave camera with respect to the master camera, using a set of time-corresponding poses of all cameras from the moving rig. While we outlined the generic pose constraints in the previous section we will now specify the model for practical purposes.

To acquire time-corresponding poses of the cameras according to the camera-local pose model described above we use a sequence of K images of each camera captured **synchronously** during an arbitrary motion of the rig. We use tracking methods based on feature points to receive estimates for the poses R_i^k, C_i^k of each camera $i = 0, \ldots, N$

with respect to its initial pose at time step 0 for each time step $k = 1, \ldots, K$ as described in section 3.5.

Resulting from the pose estimation method used we receive for each camera $i = 0, \ldots, N$ for each time step $k = 0, \ldots, K$ an estimated orientation R_i^k and position C_i^k with initial pose $\mathrm{R}_i^0 = \mathrm{I}$, $\mathrm{C}_i^0 = 0$ and $\|\mathrm{C}_i^0\| = 1$. We will continue to use an upper index to denote time and a lower index to denote the camera index where 0 refers to the master camera.

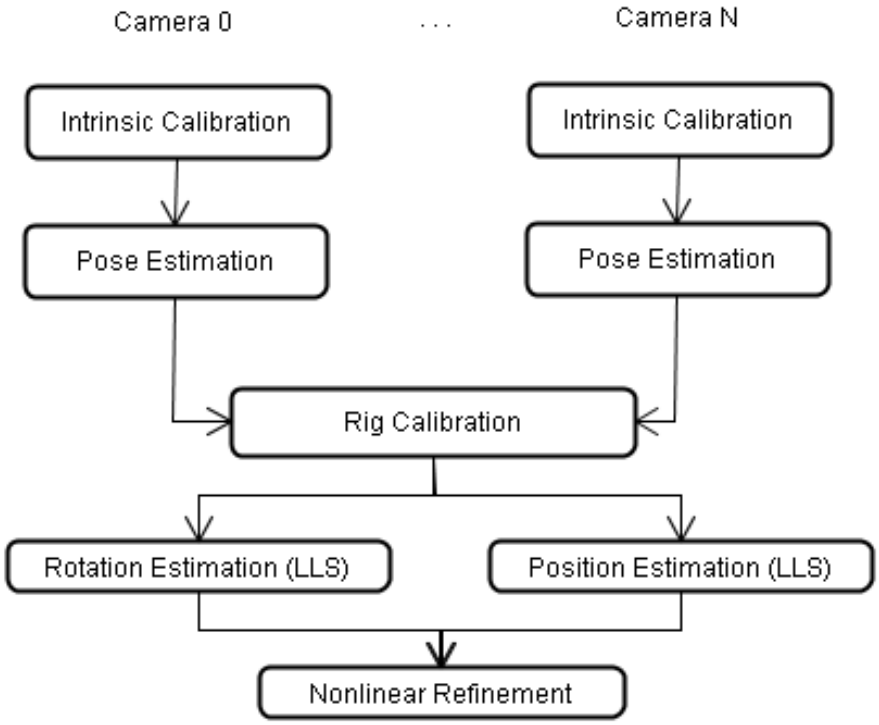


Figure 4.6.: Flowchart of the general rig calibration algorithm. First, each camera's intrinsic parameters are calibrated separately. Second, time-corresponding poses are computed for each camera. Third, decoupled linear rig calibration is performed (internal rotations first, then internal positions). Finally, combined non-linear refinement of all parameters is performed.

To register the coordinate frames of the slave cameras with the master camera we need to solve eq. (4.1) for all given time steps. As stated above, eq. (4.1) decomposes into two different sets of equations, resembling the main equation for hand-eye calibration [HD95]: Equation (4.2) regarding only orientations and eq. (4.3) combining orientations, positions, and scales. For the case of K time steps to solve two different sets of K equations for each slave camera.

According to *Horaud* [HD95], two different approaches are possible to estimate the parameters for each slave camera $i = 1 \ldots, N$:

Decoupled estimation approach Estimate both constraints subsequently:

1. First the linear constraints (4.2) are solved for the optimal internal orientation ΔR_i.

2. Afterwards the equations (4.3) are solved for internal position ΔC_i and scale $\Delta \lambda_i$ using the results of the previous estimation for ΔR_i.

This way, after solving the first constraint for rotations, the second constraint (4.3) becomes linear in position and scale. The advantage of this approach is the fact that there are only linear problems to solve.

Combined estimation approach Orientation ΔR_i, position ΔC_i and scale $\Delta \lambda_i$ are estimated simultaneously by combining both constraints. Note that the constraints (4.3) are non-linear in contrast to the first approach.

In the following part of this section we describe the decoupled estimation approach, which leads to linear least squares problems similar to the ones treated by *Horaud et al.* for hand-eye calibration [HD95]. Both the time-fixed and the time-dependent scale model will be considered.
In section 4.4 we describe the non-linear combined estimation approach which will be used to refine the solution of the linear least squares approach - it will be pointed out that combined estimation leads to better results than decoupled estimated as noticed by *Horaud* [HD95].
In section 4.5 we will modify the linear least squares method to handle also the ill-conditioned case of purely translating rig motions.
In section 4.6 the developed methods are summarized and different applications are considered.

4.3.1. Estimation of Internal Rotations

First we will consider the decoupled estimation of the internal orientation and determine the best numerical representation for orientations in this context. For a detailed discussion of different rotation representations refer to appendix A.2.

Estimation of rotations using rotation matrices

Considering orientation as 3×3 rotation matrices, the internal orientation ΔR_i for each slave camera $i = 1, \ldots, N$ can be estimated from eq. (4.2) by solving the following linear constraint:

$$R_0^k \Delta R_i = \Delta R_i R_i^k \tag{4.4}$$

With $(R_0^k \Delta R_i)_{\mu,\nu} = \sum_{\ell=1}^{3} (R_0^k)_{\mu,\ell} (\Delta R_i)_{\ell,\nu}$ and $(\Delta R_i R_i^k)_{\mu,\nu} = \sum_{\ell=1}^{3} (\Delta R_i)_{\mu,\ell} (R_i^k)_{\ell,\nu}$ for each matrix index $\mu, \nu = 1, \ldots, 3$ we get a linear equation system consisting of $9N$ unknowns and $9NK$ single equations:

$$\sum_{\ell=1}^{3} (R_0^k)_{\mu,\ell} (\Delta R_i)_{\ell,\nu} - \sum_{\ell=1}^{3} (R_i^k)_{\ell,\nu} (\Delta R_i)_{\mu,\ell} = 0 \tag{4.5}$$

for each $\mu, \nu = 1, \ldots, 3$ and $k = 1, \ldots, K$, $i = 1, \ldots, N$.

In the literature regarding rotation estimation it appears as a well-known fact that representing rotations by 3×3 rotation matrices leads to numerically instable solutions in the presence of noise. Furthermore, rotation matrices are highly redundant containing 9 parameters while each rotation in 3d space has only 3 degrees of freedom i.e. for example its Euler angles describing the amount of 3d rotation around the axes of a fixed coordinate system. We will investigate a more appropriate representation for rotations in the following paragraph.

Estimation of rotations using quaternions

There has been extensive work on solving orientation equations of the given form. Early solutions represent rotations by 3×3-rotation matrices, or alternatively 9-vectors, resulting in the linear problem formulation as in eq. (4.2). These approaches tend to be error-prone and, moreover, suffer from ensuring the orthogonality of the resulting matrices. The normalization needed to produce a numerically correct rotation matrix by orthogonalization methods such as Gram-Schmidt is computionally very expensive and introduces unnecessary opportunities for errors to propagate as stated e.g. by *Horn* in [Hor87]. Seminal contributions such as [CK91] or [HD95] represent rotations by unit quaternions which can be expressed as 4-vectors of unit length in terms of linear

algebra. In these approaches the number of parameters is reduces from 9 to 4 and the unit length constraint is far more simple to maintain than the assurance of orthonormality of rotation matrices.
In order to reduce the number of parameters for estimation of internal rotation, and to increase the stability of the estimation it has hence proved most appropriate to use unit quaternions as a representation for rotations instead of rotation matrices. For a detailed description of quaternions representing rotations see appendix A.3 or [FW04].
In the following, a solution to the rotation estimation problem is given using quaternions. It is similar to the method described by *Horaud and Dornaika* [HD95], which estimates the rotation between corresponding rotation axes using the solution for absolute orientation proposed by *Horn* [Hor87], but our approach considers the rotation amount also, as proposed by *Tsai and Lenz* [TL89], to enhance stability:

We will represent the orientation of master camera 0 and slave camera i at each time step k by unit quaternions $\mathbf{q}_0^k, \mathbf{q}_i^k$ and the internal rotation of slave camera i by a unit quaternion $\Delta\mathbf{q}_i$ instead of using rotation matrices $\mathrm{R}_0^k, \mathrm{R}_i^k, \Delta\mathrm{R}_i$.

Equation (4.2) is reformulated as an equality of quaternion products:

$$\mathbf{q}_0^k \cdot \Delta\mathbf{q}_i = \Delta\mathbf{q}_i \cdot \mathbf{q}_i^k \tag{4.6}$$

Interpreting quaternions in terms of linear algebra, eq. (4.6) gives the following linear equations for each $k = 1, \ldots, K$:

$$(\mathbf{T}_{\mathbf{q}_0^k} - \mathbf{T}^*_{\mathbf{q}_i^k})\Delta\mathbf{q}_i = 0 \tag{4.7}$$

where $\mathbf{T}_\mathbf{q}$, $\mathbf{T}^*_\mathbf{q}$ define the left and right quaternion multiplication with $\mathbf{q} = (q, x, y, z)^\mathsf{T}$ defined within linear algebra by the matrices:

$$\mathbf{T}_\mathbf{q} = \begin{pmatrix} q & -x & -y & -z \\ x & q & -z & y \\ y & z & q & -x \\ z & -y & x & q \end{pmatrix} \quad \text{and} \quad \mathbf{T}^*_\mathbf{q} = \begin{pmatrix} q & -x & -y & -z \\ x & q & z & -y \\ y & -z & q & x \\ z & y & -x & q \end{pmatrix} \tag{4.8}$$

Hence, we get from eq. (4.7) the following system of linear equations for each $\Delta\mathbf{q}_i =$

$(\Delta\mathbf{q}_i, \Delta x_i, \Delta y_i, \Delta z_i)^\mathsf{T}$ at each time step $k = 1, \ldots, K$, subject to $\|\Delta\mathbf{q}_i\| = 1$:

$$\underbrace{\begin{pmatrix} q_0^k - q_i^k & -x_0^k + x_i^k & -y_0^k + y_i^k & -z_0^k + z_i^k \\ x_0^k - x_i^k & q_0^k - q_i^k & -z_0^k - z_i^k & y_0^k + y_i^k \\ y_0^k - y_i^k & z_0^k + z_i^k & q_0^k - q_i^k & -x_0^k - x_i^k \\ z_0^k - z_i^k & -y_0^k - y_i^k & x_0^k + x_i^k & q_0^k - q_i^k \end{pmatrix}}_{=:\,\mathbf{A}_i^k} \begin{pmatrix} \Delta q_i \\ \Delta x_i \\ \Delta y_i \\ \Delta z_i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.9}$$

The equation system consists of 4 equations per pose correspondence and slave camera and 4 unknowns for each slave camera resulting in $4NK$ equations versus $4N$ unknowns. Apparently one pair of corresponding poses for each camera already suffices to estimate the internal rotation parameters by solving the arising linear equations systems (4.9) assuming that there is no noise present on the input data. From the resulting quaternions $\Delta\mathbf{q}_i$ one can compute the corresponding rotation matrices $\Delta\mathrm{R}_i$ as described in appendix A.3. Note that this method considers the vector parts of the input quaternions $\mathbf{q}_0^k, \mathbf{q}_i^k$ which correspond to the rotation axes of the rig motion, as well as the scalar quaternion part which corresponds to the amount of rotation whereas similar methods such as *Horn's* approach consider only the rotation axes.

Condition of rotation equations

In real applications, the input rotations are afflicted with noise including e.g. errors from the feature detector, matching errors and pose estimation errors. In the presence of noise, $\mathbf{A}$ generally has full rank even if the actual problem instance is ill-conditioned.
We will consider the condition of the problem first by analyzing the impact of errors on the input data to the solution. Assuming that the measured corresponding rotations $\hat{\mathbf{q}}_0^k$, $\hat{\mathbf{q}}_1^k$ are related to the true rotations by a certain error in orientation, the solution $\Delta\hat{\mathbf{q}}_i$ of the noise-afflicted equation system shall be expressed by distorting the true solution $\Delta\mathbf{q}_i$ by an error rotation $\Delta\mathbf{e}$ such that $\Delta\hat{\mathbf{q}}_i = \Delta\mathbf{e} \cdot \Delta\mathbf{q}_i$. Equation (4.6) can then be formulated as:

$$\Delta\mathbf{e}^* \cdot \hat{\mathbf{q}}_0^k \cdot \Delta\mathbf{e} = \Delta\mathbf{q}_i \cdot \hat{\mathbf{q}}_i^k \cdot \Delta\mathbf{q}_i^*$$

which can be reformulated using quaternion properties from appendix A.3 as:

$$\mathrm{R}_{\Delta\mathbf{e}}^\mathsf{T} \hat{\mathrm{r}}_0 = \mathrm{R}_{\Delta\mathbf{q}_i} \hat{\mathrm{r}}_i$$

where $\hat{\mathrm{r}}_0$, $\hat{\mathrm{r}}_i$ are the rotation axes of the measured rotations and $\mathrm{R}_{\Delta\mathbf{e}}$, $\mathrm{R}_{\Delta\mathbf{q}_i}$ are rotation matrices corresponding to the unit quaternions $\Delta\mathbf{e}$, $\Delta\mathbf{q}_i$. It appears that $\Delta\mathbf{e}$ is related

to the amount of rotation between the distorted rotation axes. Since for rotations $\mathbf{q}_0^k$, $\mathbf{q}_1^k$ close to zero-rotation the rotation axes can be distorted almost arbitrarily even by small orientation errors, the problem must be considered as ill-posed when the rig motion does not include sufficient orientation change.

From a closer examination of eq. (4.9) we see that $\mathbf{A}_i^k$ tends to degenerate for small rig rotation amounts. Consider that the rig is purely translating at one time step k, i.e. we have $\mathrm{R}_0^k = \mathrm{R}_i^k = \mathrm{I}$ or alternatively $\mathbf{q}_0^k = \mathbf{q}_i^k = (1,0,0,0)^\mathsf{T}$. In this case, we have $\mathbf{A}_i^k = 0$ and eq. (4.9) cannot be solved. In the presence of noise we find that small rig rotations within the scale of the orientation error will also lead to degenerated linear equation systems. A detailed investigation of the degenerate case of purely translating motion is presented in section 4.5. For the case of general motion we will reject motions where the rotations R_0^k or R_i^k have a rotation amount below a certain threshold $\alpha_{threshold}$ depending on the expected errors of the input rotations. This will also be explored in section 4.5, where we will compare the general motion and purely translational motion model.

Solving rotation equations

In the following, a potentially closed-form solution for the internal rotation constraints in the least squares sense will be developed, and instructions for practical solution using a singular value decomposition will be given.

In the presence of noise on the input data the system of linear equations (4.9) will not hold in general. Instead of solving the linear equations directly, the linear least squares problem corresponding to eq. (4.7) as described in section 3.6.2 is solved for a number of $K > 1$ corresponding input poses, i.e. we have to minimize the squared error function:

$$\begin{aligned}
\phi_{rot}^2(\Delta\mathbf{q}_i) &= \sum_{k=1}^{K} \|\mathbf{A}_i^k \Delta\mathbf{q}_i\|^2 \\
&= \sum_{k=1}^{K} (\mathbf{A}_i^k \Delta\mathbf{q}_i)^\mathsf{T} (\mathbf{A}_i^k \Delta\mathbf{q}_i) \\
&= \sum_{k=1}^{K} \Delta\mathbf{q}_i^\mathsf{T} \mathbf{A}_i^{k\mathsf{T}} \mathbf{A}_i^k \Delta\mathbf{q}_i \\
&= \Delta\mathbf{q}_i^\mathsf{T} (\underbrace{\sum_{k=1}^{K} \mathbf{A}_i^{k\mathsf{T}} \mathbf{A}_i^k}_{=:\mathbf{A}_i}) \Delta\mathbf{q}_i
\end{aligned} \tag{4.10}$$

subject to $\|\mathbf{q}_i\| = 1$ for each slave camera $i = 1, \ldots, N$ where each $\mathbf{A}_i$ is a symmetric 4×4 matrix. To assure unit length of solutions $\Delta\mathbf{q}_i$ there is an additional constraint $\Delta\mathbf{q}_i^\mathsf{T}\Delta\mathbf{q}_i = 1$ for each $i = 1, \ldots, N$.

We can solve this *constrained minimization problem* using Lagrangian multipliers as e.g. done by *Weng et al.* [WHA93, p.70] (see appendix A.5 for a detailed explanation). This leads to introducing an unknown scalar $\lambda \neq 0$, the *Lagrangian multiplier*, and determining the constrained minimum of ϕ_{rot}^2 as the minimum of the error function:

$$\phi_{rot}^{2*}(\Delta\mathbf{q}_i) = \Delta\mathbf{q}_i^\mathsf{T}\mathbf{A}_i\Delta\mathbf{q}_i + \lambda(1 - \Delta\mathbf{q}_i^\mathsf{T}\Delta\mathbf{q}_i)$$

with respect to $\Delta\mathbf{q}_i$. Hence, for the constrained minimum $\Delta\mathbf{q}_i$ of ϕ_{rot}^* holds:

$$\frac{\partial\phi_{rot}^{2*}}{\partial\Delta\mathbf{q}_i}(\Delta\mathbf{q}) = 2\mathbf{A}_i\Delta\mathbf{q}_i - 2\lambda\Delta\mathbf{q}_i = 0$$

and the solution can be obtained in closed form as a unit eigenvector of $\mathbf{A}$ associated with the smallest eigenvalue:

$$\mathbf{A}_i\Delta\mathbf{q}_i = \lambda\Delta\mathbf{q}_i \tag{4.11}$$

Therefore, each solution $\Delta\mathbf{q}_i$ can be obtained from eq. (4.11) by computing a unit eigenvector of $\mathbf{A}_i$ associated with the smallest positive eigenvalue. There are closed-form solutions for computing the eigenvalues and eigenvectors of a 4×4-matrix which demand to solve a fourth-order polynomial equation arising from the eigenvalue problem, such as the quartic identity originally developed by *Ferrari* in the 16th century[2]. Nevertheless, while formally a closed-form solution, it is numerical instable and far too complex to allow an analysis of error propagation. Hence, most implementors prefer iterative eigenvector decomposition methods over the solution in purely closed-form [MB99]. In our implementation a singular value decomposition of each $\mathbf{A}_i$ is used to compute the eigenvectors $\Delta\mathbf{q}_i$ as described in appendix A.1.

Absolute orientation approach for rotation estimation

In this section we will briefly address the approach for rotation estimation proposed by *Horaud and Dornaika* [HD95] by which our approach was inspired. Their approach

[2] Published by the famous Italian polymath *Girolamo Cardano* along with his seminal closed-form solution for solving general third-order polynomials in his work *"Ars Magna"* (1545) as cited in [Hor87].

reduces the rotation estimation problem to an *absolute orientation problem* regarding only rotation. This problem is defined according to *Haralick and Shapiro* by finding *"the rotation by which one or more camera reference frames can be made to correspond to a world reference frame, computed on the basis of corresponding 3d points"* [HS93]. More simple, the problem is to find the rotation that relates two sets of rays measured in different camera coordinate frames to each other. The de facto standard approach to solve the absolute orientation problem has been proposed by *Horn* in closed form [Hor87].
First we will reformulate the problem such that an absolute orientation problem is obtained, then *Horn's* solution will be outlined briefly.

It is a well-known fact from work on hand-eye calibration (q.v. [HD95]) that eq. (4.4) is equivalent to a constraint depending only on the rotation axes of the rig motion:

$$\mathrm{r}_0^k = \Delta\mathrm{R}_i \mathrm{r}_i^k \tag{4.12}$$

where r_0^k and r_i^k are the rotation axes of unit length corresponding to the rig rotations R_0^k and R_i^k.

Proof: We can transform eq. (4.4) into eq. (4.12) as follows: We know that a rotation matrix R possesses the eigenvalue 1 with the rotation axis r being the unique unit length eigenvector corresponding to 1, i.e.: $\mathrm{r} = \mathrm{Rr}$. Hence we get from eq. (4.4):

$$\begin{aligned} \mathrm{R}_0^k \Delta\mathrm{R}_i &= \Delta\mathrm{R}_i \mathrm{R}_i^k \\ \Leftrightarrow \mathrm{R}_0^k \Delta\mathrm{R}_i \mathrm{r}_i^k &= \Delta\mathrm{R}_i \mathrm{R}_i^k \mathrm{r}_i^k \\ \Leftrightarrow \mathrm{R}_0^k (\Delta\mathrm{R}_i \mathrm{r}_i^k) &= \Delta\mathrm{R}_i \mathrm{r}_i^k \\ \Leftrightarrow \mathrm{r}_0^k &= \Delta\mathrm{R}_i \mathrm{r}_i^k \end{aligned}$$

The last row follows because $\Delta\mathrm{R}_i \mathrm{r}_i^k$ appears as a unit length eigenvector of R_0^k related to the eigenvalue 1 which is uniquely defined by its rotation axis r_0^k.

■

Representing the rotations $\Delta\mathrm{R}_i$ by quaternions $\Delta\mathbf{q}_i$ as motivated above, we have the following constraints for slave camera i:

$$\mathrm{r}_0^k = \Delta\mathbf{q}_i \cdot \mathrm{r}_i^k \Delta\mathbf{q}_i^* \tag{4.13}$$

for each time step $k = 1, \ldots, K$.

Hence, the problem of estimating the internal orientation of each slave camera reduces to finding the best fitting rotation between a set of corresponding vectors. This can be done efficiently by *Horn's* closed-form solution [Hor87], which leads to solving an eigenvalue problem of a 4×4 matrix very similar to the one encountered in eq. (4.11). A detailed description of this method can be found in appendix A.4.
Nevertheless, reducing the rotation estimation problem to an absolute orientation problem yields the disadvantage that only the rotation axes are considered while the rotation amounts are ignored. As *Wang et al.* noticed in [Wan92] an additional constraint on the rotation angle yields a slight advancement of the estimation in the presence of measurement errors on the rotation amount. Therefore, rotation is estimated by our approach described in the previous section while *Horn's* method will be revisited later for rotation estimation in the case of a purely translating rig.

4.3.2. Estimation of Internal Positions

In the previous section we presented a linear method for estimating the internal rotations in the decoupled approach. Now we use the estimated internal rotations to give a system of linear equations that can be solved for internal positions and scales. First, we consider the general time-fixed scale model for convenience.

Once the internal orientations $\Delta\mathrm{R}_i$ are known[3], the internal positions $\Delta\mathrm{C}_i$ and scales $\Delta\lambda_i$ of each slave camera $i = 1, \ldots, N$ in the rig can be computed from eq. (4.3) giving the linear constraints:

$$\begin{aligned} \mathrm{R}_0^k \Delta\mathrm{C}_i + \mathrm{C}_0^k &= \Delta\lambda_i \Delta\mathrm{R}_i \mathrm{C}_i^k + \Delta\mathrm{C}_i \\ \Leftrightarrow \quad (\mathrm{I} - \mathrm{R}_0^k)\Delta\mathrm{C}_i + \Delta\mathrm{R}_i \mathrm{C}_i^k \Delta\lambda_i &= \mathrm{C}_0^k \end{aligned} \tag{4.14}$$

for each time step $k = 1, \ldots, K$.

From eq. (4.14) we get the following system of linear equations:

$$\underbrace{\begin{pmatrix} \mathrm{I} - \mathrm{R}_0^k & \Delta\mathrm{R}_i \mathrm{C}_i^k \end{pmatrix}}_{=: \mathbf{B}_i^k} \begin{pmatrix} \Delta\mathrm{C}_i \\ \Delta\lambda_i \end{pmatrix} = \mathrm{C}_0^k \tag{4.15}$$

for each time step $k = 1, \ldots, K$.

[3] Throughout this section the rotation matrix and quaternion representations for rotations will be used interchangeably. See app. A.3 for the conversion between both representations.

The equation system (4.14) consists of 3 equations per pose correspondence and slave camera and 4 unknowns for each slave camera resulting in $3NK$ equations versus $4N$ unknowns. Hence at least $K = 2$ corresponding pose pairs for each slave camera are necessary for a unique solution, which requires two independent motions of the rig.

Note that the estimated position is measured within the master camera reference frame. In order to achieve metric calibration consistent with the real world's metric one has to determine the scale λ_{world} between the world coordinate frame and the master camera reference frame also.
To do so, it is unavoidable to use a calibration object of known geometry and size for the image sequence recorded by the master camera. If we are able to determine the length of some vectors in views of the master camera with sufficient accuracy according to the world's metric, we are able to deduce the scale λ_{world}.

Condition of position equations

First note that the system of linear equations (4.15) is degenerate when there is no significant rotation R_0^k of the rig present for some time step k or if there is no translation C_i^k of the slave camera. The latter occurs only when there has been no motion at all or if the rig has been rotated around the slave camera center. This case will not be investigated. The first case is referred to as *purely translational motion*. We will consider it in section 4.5. For the case of an arbitrarily moving rig we will reject motions where the rotation amount α_0^k is below a certain threshold $\alpha_{threshold}$ as stated for rotation estimation.
Second, since rotation estimation is performed separately, estimation errors of $\Delta\mathrm{R}_i$ are transferred to position and scale estimation.
Finally, solving the equation system for position and scale by a linear least squares approach in the presence of measurement errors on the input poses does not impose bounds on $\Delta\mathrm{C}_i$ and $\Delta\lambda_i$ explicitly. Strategies for further constraining $\Delta\lambda_i$ to counter deviations of the estimates are given in the next section.

Solving position equations

In the presence of noise we consider the linear least squares problem corresponding to eq. (4.7) as described in section 3.6.2, i.e. we minimize the squared error function

$$\phi_{pos}^2(\Delta \mathrm{C}_i, \Delta\lambda_i) = \sum_{k=1}^{K} (\mathbf{B}_i^k \begin{pmatrix} \Delta \mathrm{C}_i \\ \Delta\lambda_i \end{pmatrix} - \mathrm{C}_0^k)^2 = \| \begin{pmatrix} \mathbf{B}_i^1 \\ \vdots \\ \mathbf{B}_i^K \end{pmatrix} \begin{pmatrix} \Delta \mathrm{C}_i \\ \Delta\lambda_i \end{pmatrix} - \begin{pmatrix} \mathrm{C}_0^1 \\ \vdots \\ \mathrm{C}_0^K \end{pmatrix} \|^2 \quad (4.16)$$

for each slave camera $i = 1, \ldots, N$ using singular value decomposition as described in appendix A.1.

Estimation with time-dependent scales

Finally, the linear least squares estimation of internal positions and scales is extended from the time-fixed to the time-dependent scale model, i.e. we assume that for each i-th camera n individual internal scales $\Delta\lambda_i^{(\nu)}$ with $\nu = 1, \ldots, n$ are defined, each valid for a subset of equations (4.14) with $k \in \mathcal{K}^{(\nu)}$ where $\mathcal{K}^{(1)}, \ldots, \mathcal{K}^{(n)}$ is a partition of time steps $\{1, \ldots, K\}$ as defined in section 4.2.2.

Hence, the equation system stated above is generalized to:

$$(\mathrm{I} - \mathrm{R}_0^k)\Delta \mathrm{C}_i + \Delta \mathrm{R}_i \mathrm{C}_i^k \Delta\lambda_i^{(\nu)} = \mathrm{C}_0^k \quad (4.17)$$

where ν denotes the respective time set with $k \in \mathcal{K}^{(\nu)}$ for each $k = 1, \ldots, K$. Note that we have $3+n$ unknowns for each slave camera resulting in $3NK$ equations versus $(3+n)N$ unknowns. Since $n \leq K$ even for the worst case of maximal number of unknowns, $K = 2$ single movements of the rig suffice to yield an non-underconstrained equation system.

4.3.3. Separate Estimation of Internal Scales

As we have seen, eq. (4.3) provides constraints on the internal positions $\Delta \mathrm{C}_i$ and scales $\Delta\lambda_i$ of each camera in the rig after estimating the internal orientations $\Delta \mathrm{R}_i$.

In the previous section we solved the resulting linear equations system eq. (4.15) for internal positions and scales simultaneously. As stated in eq. (4.15), internal positions and scales are theoretically unbounded, leading to potentially severe estimation errors in the presence of noise on the measured corresponding poses. Nevertheless, there are further constraints on the internal scale, depending only on the measured poses, that have to be investigated in order to constrain the internal scale appropriately. The following constraint holds for each slave camera $i = 1, \ldots, N$ at each time step $k = 1, \ldots, K$:

$$\Delta\lambda_i {\mathrm{C}_i^k}^{\mathsf{T}} \mathrm{r}_i^k = {\mathrm{C}_0^k}^{\mathsf{T}} \mathrm{r}_0^k \tag{4.18}$$

where C_0^k, C_i^k are the positions of the master and slave camera measured in their respective local coordinate frame $\mathcal{C}^i$, and r_0^k, r_i^k are the rotation axes of master and slave rotations R_0^k, R_i^k.

Proof: Assume that the positions C_0, C_i and orientations R_0, R_i of master and slave camera measured in their respective local coordinate frames are given for a certain point of time. We will omit the superscript k for the sake of legibility. The rig parameters for the slave camera are given by $\Delta\mathrm{C}_i$, $\Delta\mathrm{R}_i$ and scale $\Delta\lambda_i$. Equation (4.3) provides the following constraint:

$$(\mathrm{R}_0 - \mathrm{I})\Delta\mathrm{C}_i + \mathrm{C}_0 = \Delta\lambda_i \Delta\mathrm{R}_i \Delta\mathrm{C}_i \tag{4.19}$$

From eq. (4.12) we obtain the following constraint for the rotation axes r_0, r_1 of master and slave rotations R_0, R_1:

$$\mathrm{r}_0 = \Delta\mathrm{R}_i \mathrm{r}_1 \tag{4.20}$$

Also, it can be verified easily that the vector $(\mathrm{R}_0 - \mathrm{I})\Delta\mathrm{C}_i$ is perpendicular to the rotation axis r_0 of the master camera:

$$\begin{aligned} (\mathrm{R}_0 - \mathrm{I})\Delta\mathrm{C}_i^{\mathsf{T}} \mathrm{r}_0 &= \mathrm{R}_0 \Delta\mathrm{C}_i^{\mathsf{T}} \mathrm{r}_0 - \Delta\mathrm{C}_i^{\mathsf{T}} \mathrm{r}_0 \\ &= \Delta\mathrm{C}_i^{\mathsf{T}} \underbrace{\mathrm{R}_0 \mathrm{r}_0}_{=\mathrm{r}_0} - \Delta\mathrm{C}_i^{\mathsf{T}} \mathrm{r}_0 = 0 \end{aligned} \tag{4.21}$$

Therefore, we now have:

$$\begin{aligned}
\Delta\lambda_i \mathrm{C}_i{}^{\mathsf{T}}\mathrm{r}_i &= \Delta\lambda_i \mathrm{C}_i{}^{\mathsf{T}}(\Delta\mathrm{R}_i^{\mathsf{T}}\Delta\mathrm{R}_i)\mathrm{r}_i \\
&= (\underbrace{\Delta\lambda_i\Delta\mathrm{R}_i\mathrm{C}_i}_{=(\mathrm{R}_0-\mathrm{I})\Delta\mathrm{C}_i+\mathrm{C}_0})^{\mathsf{T}}(\underbrace{\Delta\mathrm{R}_i\mathrm{r}_i}_{=\mathrm{r}_0}) \\
&\overset{(4.19,4.20)}{=} ((\mathrm{R}_0-\mathrm{I})\Delta\mathrm{C}_i+\mathrm{C}_0)^{\mathsf{T}}\mathrm{r}_0 \\
&= \underbrace{(\mathrm{R}_0-\mathrm{I})\Delta\mathrm{C}_i^{\mathsf{T}}\mathrm{r}_0}_{=0}+\mathrm{C}_0^{\mathsf{T}}\mathrm{r}_0 \\
&\overset{(4.21)}{=} \mathrm{C}_0^{\mathsf{T}}\mathrm{r}_0
\end{aligned} \tag{4.22}$$

■

Given measured poses at time step k, the scale $\Delta\lambda_i^k$ computed from eq. (4.18) by $\Delta\lambda_i^k = \frac{\mathrm{C}_0^{k\mathsf{T}}\mathrm{r}_0^k}{\mathrm{C}_i^{k\mathsf{T}}\mathrm{r}_i^k}$ is a property of measurements only; hence, each such $\Delta\lambda_i^k$ can be interpreted as a *measurement for internal scale* at time step k. To distinguish measured scales from estimated scales the former will be denoted simply by λ_i^k in the following.

Condition of scale equations

Equation (4.18) degenerates for motions k where $\mathrm{C}_i^{k\mathsf{T}}\mathrm{r}_i^k$ approaches zero. This situation occurs for motions where a slave camera translates mainly perpendicular to its rotation axis, or if there is either no significant translation or rotation of the respective slave camera noticeable. To prevent degeneracy, such motions have to be rejected for scale estimation. If $\mathrm{C}_i^{k\mathsf{T}}\mathrm{r}_i^k$ evaluates below a certain threshold $\delta_{threshold}$, a motion is considered as degenerate. We will find that a good threshold is $\delta_{threshold} = 3\varepsilon$, where ε denotes the absolute error expected on either scalar product $\mathrm{C}_0^{k\mathsf{T}}\mathrm{r}_0^k, \mathrm{C}_i^{k\mathsf{T}}\mathrm{r}_i^k$.

Remark: Denote both scalar products as $s_0 = \mathrm{C}_0^{k\mathsf{T}}\mathrm{r}_0^k$, $s_i = \mathrm{C}_i^{k\mathsf{T}}\mathrm{r}_i^k$ for simplicity. Suppose that both s_0 and s_i have an error of magnitude $\varepsilon > 0$. The worst-case absolute error for $\hat{\lambda}_i^k = \frac{\hat{s}_0}{\hat{s}_i}$ is given by $\varepsilon_\lambda = \hat{\lambda}_i^k - \lambda_i^k = \frac{s_0+e}{s_i-e} - \frac{s_0}{s_i}$. By determining a relative error factor $L > 0$ such that the absolute error on $\hat{\lambda}_i^k$ is bounded by $L\lambda_i^k$ we get the

following constraint:

$$\begin{array}{rcrcl} \varepsilon_\lambda & = & \frac{s_0+\varepsilon}{s_i-\varepsilon} - \frac{s_0}{s_0} & < & L\frac{s_0}{s_0} \\ & \Leftrightarrow & \frac{s_0+\varepsilon}{s_i-\varepsilon} & < & (L+1)\frac{s_0}{s_i} \\ & \Leftrightarrow & s_0 s_i + s_i\varepsilon & < & (L+1)(s_0 s_i - s_0\varepsilon) \\ & \Leftrightarrow & s_i & > & \frac{L+1+s_i/s_0}{L}\varepsilon > \frac{L+1}{L}\varepsilon \end{array} \tag{4.23}$$

With $L = 0.5$, we have $\delta_{threshold} = 3\varepsilon$, where ε is appropriately considered to be within the range of 10^{-3}.

■

The measured scale λ_i^k is in general very sensitive to noise on the measured poses since it is computed as a ratio of measurements afflicted with independent errors.
In the time-fixed scale model a linear least squares approach can be used over all measured scales $\lambda_i^1, \ldots, \lambda_i^K$, since the rig scale $\Delta\lambda_i$ is assumed to be constant. For the time-dependent scale model we assume n different scales for different subsets of $\{1, \ldots, K\}$. As for the time-fixed scale model the rig scale $\Delta\lambda_i^{(\nu)}$ for each time set $\mathcal{K}^{(\nu)}$ can be estimated by a linear squares approach with respect to the measured scales from $\mathcal{K}^{(\nu)}$. In the worst case, namely $n = K$, each $\Delta\lambda_i^{(k)}$ is identified with the measured scale λ_i^k without potential optimization.

Solving scale equations

Given K noise-afflicted motions of the rig, $\Delta\lambda_i$ is estimated for each slave camera $i = 1, \ldots, N$ in the time-fixed scale model from eq. (4.18) by minimizing the error between $\Delta\lambda_i \mathrm{C}_i^{k\mathsf{T}} \mathrm{r}_i^k$ and $\mathrm{C}_0^{k\mathsf{T}} \mathrm{r}_0^k$ in the least squares sense, which leads to minimizing the squared error function:

$$\phi^2_{scale}(\Delta\lambda_i) = \sum_{k=1}^{K} (\mathrm{C}_i^{k\mathsf{T}} \mathrm{r}_i^k \Delta\lambda_i - \mathrm{C}_0^{k\mathsf{T}} \mathrm{r}_0^k)^2 = (\mathrm{a}^\mathsf{T} \Delta\lambda_i - \mathrm{b})^2 \tag{4.24}$$

for each slave camera i, where $\mathrm{a}, \mathrm{b} \in \mathbb{R}^K$ are given by $a_k := \mathrm{C}_i^{k\mathsf{T}} \mathrm{r}_i^k$ and $b_k := \mathrm{C}_0^{k\mathsf{T}} \mathrm{r}_0^k$ for each $k = 1, \ldots, K$. The linear least squares estimator as defined in 3.6.2 is given in closed form by:

$$\Delta\hat{\lambda}_i = \frac{\mathrm{a}^\mathsf{T}\mathrm{b}}{\mathrm{a}^\mathsf{T}\mathrm{a}} \tag{4.25}$$

To estimate each internal scale $\Delta\lambda_i^{(\nu)}$ for the ν-th time set $\mathcal{K}^{(\nu)} \subseteq \{1, \ldots, K\}$, the least squares estimator regards measurements for $k \in \mathcal{K}^{(\nu)}$ only. For time sets of size 1, no optimization is possible.

Recapitulating, we have proposed a solution for the internal scales depending on the measured camera poses both for the time-dependent and time-fixed scale mode. Solutions from this approach are considered as measurements of the internal scale and will be integrated into the following non-linear refinement of all rig parameters as a constraint on the respective parameters.

4.4. Non-Linear Refinement of Rig Parameters

In the previous section the estimation was performed by solving systems of linear equations first for internal rotation and afterwards for internal position and scale. It is obvious that errors in the estimation of internal rotation will impact the estimation of the internal translation and scale. Moreover, we pointed out that there is additional knowledge about measurements of the internal scales accessible which can be used to impose further constraints on the rig parameters.
Horaud suggests in [HD95] on the topic of hand-eye calibration that all parameters should be estimated simultaneously from eq. (4.6) and eq. (4.14) by combining the error functionals eq. (4.10) (rotation) and eq. (4.16) (position and scale) while noticing that there has been found no closed-form solution of this non-linear problem yet.
While for the decoupled linear rotation estimation the unit length constraint $\|\Delta\mathbf{q}_i\| = 1$ for each internal rotation quaternion is maintained implicitly, it has to be added explicitly for non-linear refinement. A common technique to add such constraints is to add a penalty term for $\|\Delta\mathbf{q}_i\|$ deviating from 1 to the error functional.

The non-linear error functional reads therefore:

$$\begin{aligned}\phi^2_{NL}(\Delta\mathbf{q}_i, \Delta\mathrm{C}_i, \Delta\lambda_i) &= \phi^2_{rot}(\Delta\mathbf{q}_i) + \phi^2_{pos}(\Delta\mathbf{q}_i, \Delta\mathrm{C}_i, \Delta\lambda_i) + \xi_1(\|\Delta\mathbf{q}_i\| - 1)^2 \\ &= \sum_{k=1}^{K} \|\mathbf{q}_0^k \cdot \Delta\mathbf{q}_i - \Delta\mathbf{q}_i \cdot \mathbf{q}_i^k\|^2 + \\ &\quad \sum_{k=1}^{K} \|(\mathrm{R}_0^k - \mathrm{I})\Delta\mathrm{C}_i + \Delta\lambda_i \Delta\mathbf{q}_i \cdot \mathrm{C}_i^k \cdot \Delta\mathbf{q}_i^* - \mathrm{C}_0^\kappa\|^2 + \\ &\quad \xi_1(\Delta\mathbf{q}_i^\mathsf{T}\Delta\mathbf{q}_i - 1)^2\end{aligned} \tag{4.26}$$

where ξ_1 defines a considerably large weight on the penalty term enforcing unit length for each internal rotation quaternion. It is set to $\xi_1 = 2 \cdot 10^6$ as suggested in [HD95].

This error functional has the form of a sum of squares of non-linear functions and is considered as a classical non-linear least squares problem (see section 3.6.3). In our implementation the problem is solved using a Levenberg-Marquardt non-linear optimization method as described in [PFTW88]. The optimization method starts with the

solutions $\Delta\mathbf{q}_i^0, \Delta\mathrm{C}_i^0, \Delta\lambda_i^0$ obtained from the linear least squares method, refining the solution iteratively. Note that the estimated rotation may be inferior than solutions from decoupled rotation estimation since it is additionally afflicted with position measurement errors. Nevertheless, since the error functional depends on all rig parameters at the same time, it is suggested that in general the minimization converges to a better solution. We will evaluate the performance of the refinement in section 5.1.

Additional scale measurement constraints

In section 4.3.3 we derived further properties of the internal scales $\Delta\lambda_i$. The solutions λ_i of the linear least squares problem (4.24) can be interpreted as measurements of the internal scales arising from the measured poses. Uncertainty proportions of the measurements are given by their standard deviations $\sigma_i^2 = \sum_{k=1}^{K}(\lambda_i - \lambda_i^k)$ where $\lambda_i^k = \frac{\mathrm{C}_0^k \mathrm{r}_0^k}{\mathrm{C}_i^k \mathrm{r}_i^k}$ for each $k = 1, \ldots, K$.
The error functional (4.26) is modified in order to constrain the solution to fit the measured scales considering their respective uncertainties by applying a penalty term for the additional implicit constraint $\Delta\lambda_i - \lambda_i = 0$ weighted each with $1/\sigma_i^2$.

The modified non-linear error functional is then given by:

$$\phi_{NL}^{*2}(\Delta\mathbf{q}_i, \Delta\mathrm{C}_i, \Delta\lambda_i) = \phi_{NL}^2(\Delta\mathbf{q}_i, \Delta\mathrm{C}_i, \Delta\lambda_i) + \frac{(\Delta\lambda_i - \lambda_i)^2}{\sigma_i^2} \tag{4.27}$$

This approach is extended straightforwardly for the time-dependent scale model by adding a term for each internal scale $\Delta\lambda_i^{(\nu)}$ with respect to the measurement $\lambda_i^{(nu)}$ of time set $\mathcal{K}^{(\nu)}$.

4.5. Estimation with a Purely Translating Rig

As stated in sections 4.3.1 and 4.3.2, the systems of linear equations for decoupled estimation of rotation (eq. (4.7)) and position and scale (eq. (4.15)) tend to degenerate in the absence of significant rotation of the rig. More precisely: When the rig motion is purely translational, the relative orientations $\mathbf{q}_0^k$ and $\mathbf{q}_i^k$ in eq. (4.6) are both given by the quaternion representing zero rotation $(1, 0, 0, 0)^\mathsf{T}$ and the matrix $\mathbf{A}_i^k$ in eq. (4.7) becomes zero. Even when the rotation of the rig is very small, $\mathbf{A}_i^k$ is close to zero and the equation system eq. (4.7) becomes ill-conditioned in the presence of noise.

This is also visible in eq. (4.15) where $\mathbf{B}_i^k$ grows ill-conditioned when the rotation R_0^k observed by the master camera is close to I.

To consider such motions we introduced the purely translational motion model (PTM) in section 4.2.1. To decide if the given motion fits the PTM we are looking at the input pose orientations R_i^k. Given that all pose orientations are below a certain minimal rotation threshold α_{maxrot} we consider the rig to be purely translating. This case has been referred to as *purely translation motion model* in opposition to the *general motion model* where sufficient rotation is present for each frame. For the general motion model poses with orientation below α_{maxrot} are excluded from estimation. An appropriate threshold is determined in section 5.2.

Obviously the estimation of the $\Delta\mathrm{C}_i$ is not possible for the case of a purely or almost entirely translating rig, since pure translation yields no constraints on the relative positions of the coupled cameras. Motion measured within the camera-local coordinate frames appears equally for a purely translational motion regardless of where the frames are located within the world coordinate frame. However, internal orientation and scale can still be estimated.

Estimation of internal rotations: Assuming that the input rotations R_i^k are each equal to I, eq. (4.14) turns into:

$$\Delta\mathrm{R}_i\Delta\lambda_i\mathrm{C}_i^k = \mathrm{C}_0^k \tag{4.28}$$

Hence, the problem of estimating the internal orientation of each slave camera reduces to finding the best fitting rotation between a set of corresponding vectors, i.e. by solving an *absolute orientation problem* regarding only rotation as defined in section 4.3.1: “Absolute orientation approach for rotation estimation”. Denoting the directions of the position vectors as $\mathrm{c}_i^k = \mathrm{C}_i^k/\|\mathrm{C}_i^k\|$, eq. (4.28) can be reduced to estimate $\Delta\mathrm{R}_i$ without knowledge about $\Delta\lambda_i$:

$$\Delta\mathrm{R}_i\mathrm{c}_i^k = \mathrm{c}_0^k \tag{4.29}$$

In the presence of normally distributed noise, an optimal solution is found by minimizing the squared error function:

$$\phi_{rot}^2(\Delta\mathrm{R}_i) = \sum_{k=1}^{K} \|\Delta\mathrm{R}_i\mathrm{c}_i^k - \mathrm{c}_0^k\|^2 \tag{4.30}$$

This can be solved efficiently according to *Horn's* closed-form solution for absolute orientation [Hor87] which was discussed above (see appendix A.4 for further details).

Estimation of internal scales: As noted in section 4.3.3, the scale estimation from input poses also fails for a purely or almost entirely translating rig. However, since a rotation does not change the length of a vector, the scale $\Delta\lambda_i$ can be estimated directly from eq. (4.28) without knowledge about $\Delta\mathrm{R}_i$. For each time step k holds:

$$\Delta\lambda_i \|\Delta\mathrm{R}_i \mathrm{C}_i^k\| = \|\mathrm{C}_0^k\| \Leftrightarrow \|\mathrm{C}_i^k\| \Delta\lambda_i = \|\mathrm{C}_0^k\| \tag{4.31}$$

Equation (4.31) provides measurements $\lambda_i^k = \frac{\|\mathrm{C}_0^k\|}{\|\mathrm{C}_i^k\|}$ of the internal scale for each time step k. Similar to scale estimation for the general motion model, the linear least square estimator for each $\Delta\lambda_i$ in the time-fixed scale model minimizes the squared error function:

$$\phi^2_{scale}(\Delta\lambda_i) = \sum_{k=1}^{K} (\|\mathrm{C}_i^k\| \Delta\lambda_i - \|\mathrm{C}_0^k\|)^2 = (\mathrm{a}^\mathsf{T} \Delta\lambda_i - \mathrm{b})^2 \tag{4.32}$$

and is given in closed-form by $\Delta\hat{\lambda}_i = \frac{\mathrm{a}^\mathsf{T}\mathrm{b}}{\mathrm{a}^\mathsf{T}\mathrm{a}}$ for each slave camera i where $\mathrm{a}, \mathrm{b} \in \mathbb{R}^K$ are given by $a_k := \|\mathrm{C}_i^k\|$ and $b_k := \|\mathrm{C}_0^k\|$ for each $k = 1, \ldots, K$. The linear least squares estimation is generalized to the time-dependent case the same way as suggested for the general motion model by estimating a solution for each time set respectively.

This gives us a strategy to estimate internal rotations and scales from measured corresponding camera poses for the purely translational motion model. We will compare this approach with the general motion model approach in section 5.2 for rig motions with small amounts of rotation and give a method to select the preferable model.

4.6. Closure

In this chapter we have developed a framework for rig calibration from time-corresponding poses of the coupled cameras. Intrinsic parameters and internal pose parameters are calibrated separately where standard techniques depending on the specific camera model are used for the former task. We presented two approaches for internal pose estimation: A linear approach which estimates rotation and position decoupled, and a non-linear approach where all parameters are estimated at simultaneously. Non-linear estimation of internal scales between the rig camera reference frames is further constrained by scale estimates computed from corresponding poses.
We stated analytically that both approaches cannot be applied for the purely translational motion model (PTM), i.e. for rig motions without significant rotation with respect to the reference pose. A modification of the internal pose calibration approach for the PTM was given where internal rotations and scales can be recovered while internal positions are not accessible.

Finally, modifications of position estimation for general rig motions assuming a time-dependent scale model (TDS) were given, i.e. considering that the scales between the rig camera reference frames are not equal for all corresponding poses.
Figure 4.7 shows an overview of the internal pose estimation variants for all motion and scale models. In the next section, an implementation of the approaches will be evaluated in practical tests considering first synthetic data for stability analysis, then real applications.

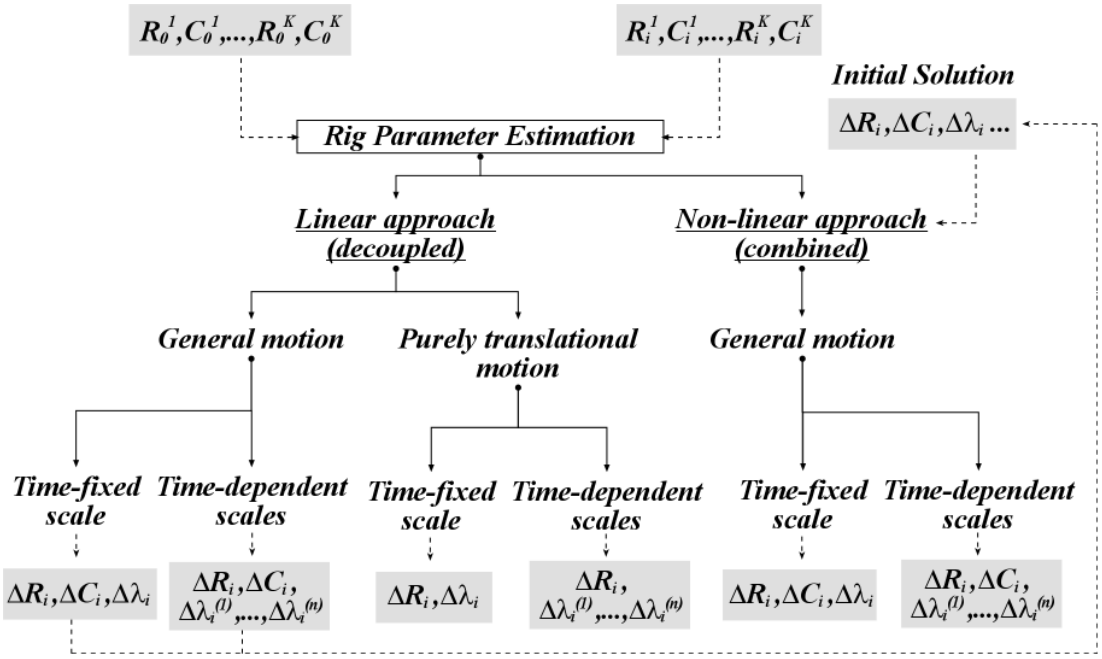


Figure 4.7.: Overview of different versions of our rig estimation approach for dedicated problems. The appropriate method instance depends on the selection of general rig motion model vs. purely translational rig motion model, and on the proposition of time-fixed internal scales $\Delta\lambda_i$ vs. time-dependent internal scales $\Delta\lambda_i^{(\nu)}$.
Input are corresponding poses of master 0 and each slave camera i at certain time steps $k = 1, \ldots, K$. The leafs correspond to the output values of the estimation algorithm i.e. internal rotation $\Delta\mathrm{R}_i$, positions $\Delta\mathrm{C}_i$ and time-fixed scale $\Delta\lambda_i$ or alternatively time-dependent scales $\Delta\lambda_i^{(\nu)}$ for each slave camera i. The non-linear estimation method requires an initial solution which can be computed by the linear least squares method for the general motion model.

5. Applications and Results

In this chapter our approach is evaluated in practice using different test cases.

In the following section we will first evaluate the stability of the implementation using synthetic data. For this purpose, appropriate error models for pose estimation and rig calibration are introduced. Then we will present different real applications, evaluate the quality of the estimated rig parameters, and demonstrate how the proposed calibration approaches can be applied practically.

5.1. Stability Analysis

In this section the stability of the implemented rig calibration methods against noise on the input data is evaluated. We use a large sample set of synthetically generated pose correspondences afflicted with different amounts of error in order to determine the robustness of the linear least squares calibration method and the non-linear calibration method with respect to the number of input poses and input error amount. The special case of purely translational rig motion as described in section 4.5 will also be evaluated by comparing the rig calibration methods for both motion models using synthetic pose correspondences where the rig is only rotated by very small amounts.

5.1.1. Error Model

To measure the error of the estimated rig parameters $\Delta\hat{\mathrm{R}}_i$, $\Delta\hat{\mathrm{C}}_i$ for each slave camera $i = 1, \ldots, N$ with respect to the real parameters, the following descriptive error model is used as suggested in [HD95]: Errors of orientations are measured by the rotation between expected and measured orientation. For positions, both the orientation error of the position direction vector and the length error of the position vector are considered respectively.
The same error model is used to describe the error of time-depending input poses $\hat{\mathrm{R}}_i^k$, $\hat{\mathrm{C}}_i^k$, e.g. provided by pose estimation, with respect to the true poses R_i^k, C_i^k for each

camera $i = 0, \ldots, N$ at time step $k = 1, \ldots, K$.
The error model is extended by the internal rig scale $\Delta\lambda_i$ and pose-corresponding scales λ_i^k (i.e. "measured scales").

Rotation orientation error The *rotation orientation error* ε_{R} measures the error of an estimated rotation $\hat{\mathrm{R}}$. It is defined by the amount of rotation that remains from a rotation by R and subsequent back-rotation by $\hat{\mathrm{R}}^{\mathsf{T}}$. Representing rotations by unit quaternions, the amount of error rotation is given by $\varepsilon_{\mathrm{R}} = 2\arccos(\varepsilon_q)$ where $(\varepsilon_q, \varepsilon_x, \varepsilon_y, \varepsilon_z)^{\mathsf{T}} = \hat{\mathbf{q}}^* \cdot \mathbf{q}$ (see appendix A.3).

Position orientation error The *position orientation error* ε_{C} measures the error of the direction of an estimated position vector $\hat{\mathrm{C}}$. It is defined by the angle between C and $\hat{\mathrm{C}}$ which can be computed by $\varepsilon_{\mathrm{C}} = \arccos(\frac{\mathrm{C}^{\mathsf{T}}\hat{\mathrm{C}}}{\|\mathrm{C}\|\|\hat{\mathrm{C}}\|})$.

Distance error The *distance error* $\varepsilon_{\|\mathrm{C}\|}$ measures the error of the length of an estimated position vector $\hat{\mathrm{C}}$. It is given relative to the real length by $\varepsilon_{\|\mathrm{C}\|} = |\frac{\|\mathrm{C}\| - \|\hat{\mathrm{C}}\|}{\|\mathrm{C}\|}|$ in percentage of $\|\mathrm{C}\|$.

Scale error The scale error ε_λ of an estimated scalar $\hat{\lambda}$ is given relative to the real scale λ by $\varepsilon_\lambda = |\frac{\lambda - \hat{\lambda}}{\lambda}|$ in percentage of λ.

To evaluate the quality of corresponding poses from pose estimation for rig parameter estimation we also consider the amount by which the poses violate the rig constraints eq. (4.2) and eq. (4.3) regarding the real rig parameters. Given estimated time-corresponding poses $\hat{\mathrm{R}}_0^k$, $\hat{\mathrm{C}}_0^k$ and $\hat{\mathrm{R}}_i^k$, $\hat{\mathrm{C}}_i^k$ the expected slave pose is computed from the estimated master pose using the real rig parameters $\Delta\mathrm{R}_i$, $\Delta\mathrm{C}_i$, $\Delta\lambda_i$ as stated below. The deviations of eq. (4.2) and eq. (4.3) from equality, i.e. the *constraint residua*, are used to assess the validity of the estimated corresponding poses with respect to their value for rig parameter estimation.

Rotation constraint residuum Given estimated corresponding rotations $\hat{\mathrm{R}}_0^k$ and $\hat{\mathrm{R}}_i^k$ the *rotation constraint orientation residuum* of eq. (4.2) is given by the orientation error on both sides of eq. (4.2) between $\hat{\mathrm{R}}_0^k\Delta\mathrm{R}_i$ and $\Delta\mathrm{R}_i\hat{\mathrm{R}}_i^k$ for each time step $k = 1, \ldots, K$.

Position constraint residuum Given estimated corresponding poses $\hat{\mathrm{R}}_0^k$, $\hat{\mathrm{C}}_0^k$ and $\hat{\mathrm{R}}_i^k$, $\hat{\mathrm{C}}_0^k$ the *position constraint orientation residuum* is given by the orientation error between both hands of eq. (4.3), i.e. between vectors $\hat{\mathrm{R}}_0^k\Delta\mathrm{C}_i + \hat{\mathrm{C}}_0^k$ and $\Delta\lambda_i\Delta\mathrm{R}_i\hat{\mathrm{C}}_i^k + \Delta\mathrm{C}_i$.
The *absolute position constraint residuum* is given by the absolute difference between both equation hands $\|(\hat{\mathrm{R}}_0^k - \mathrm{I})\Delta\mathrm{C}_i + \hat{\mathrm{C}}_0^k - \Delta\lambda_i\Delta\mathrm{R}_i\hat{\mathrm{C}}_i^k\|$.

Scale constraint residuum Given estimated corresponding poses $\hat{\mathrm{R}}_0^k$, $\hat{\mathrm{C}}_0^k$ and $\hat{\mathrm{R}}_i^k$, $\hat{\mathrm{C}}_0^k$ the *scale constraint error* is given by the scale error of "measured scales" $\hat{\lambda}_i^k$ computed by eq. (4.18) with respect to the real scales $\Delta\lambda_i^k$ between both camera reference frames at time step $k = 1, \ldots, K$ (remember that for the time-fixed scale model $\Delta\lambda_i^k$ is assumed to be constant for all k, denoted by $\Delta\lambda_i$), i.e. $|\frac{\Delta\lambda_i^k - \hat{\lambda}_i^k}{\Delta\lambda_i^k}|$.

5.1.2. Stability Depending on Pose Error

In the first test case we analyze and compare the stability of the estimation methods proposed in **Chapter 4**. We do so by applying the methods to a large number of randomly generated input pose correspondences corrupted by artificial noise. This procedure is applicable because each single estimation can be performed with very low time effort.

First, we create random rig parameters $\Delta\mathrm{R}_i$, $\Delta\mathrm{C}_i$, $\Delta\lambda_i$ for each slave camera $i = 1, \ldots, N$, i.e. the internal poses and scales of each slave camera with respect to the master camera.

- Each $\Delta\mathrm{C}_i$ is uniform randomly chosen from the subspace $[-10, 10]^3$.
- For each rotation $\Delta\mathrm{R}_i$ a unit vector r is uniform randomly chosen from the unit sphere and an angle α is uniform randomly selected from the interval $[-\pi, \pi]$. The rotation is composed from axis r and angle α.
- Each scale $\Delta\lambda_i$ is uniform randomly chosen from the interval $[0.01, 1]$.

Next, we synthetically generate poses from a random motion of the rig over K frames.

- At first, random poses R_0^k, C_0^k are generated for the master camera 0 for each time step $k = 1, \ldots, K$.
- Afterwards, the corresponding pose parameters R_i^k, C_i^k for each slave camera $i = 1, \ldots, N$ are computed by applying the rig parameters $\Delta\mathrm{R}_i$, $\Delta\mathrm{C}_i$, $\Delta\lambda_i$ to the master camera poses. The computation of slave poses from master poses and rig parameters can be derived from eq. (4.2) and eq. (4.3):

$$\mathrm{R}_i^k = \Delta\mathrm{R}_i^\mathsf{T} \mathrm{R}_0^k \Delta\mathrm{R}_i \tag{5.1}$$

and:

$$\mathrm{C}_i^k = \frac{1}{\Delta\lambda_i}\Delta\mathrm{R}_i^{\mathsf{T}}((\mathrm{R}_0^k - \mathrm{I})\Delta\mathrm{C}_i + \mathrm{C}_0^k) \tag{5.2}$$

for each time step $k = 1, \ldots, K$.

In order to simulate the uncertainty of the master and slave poses obtained by previous pose estimation techniques, all input poses are disturbed by a certain error amount. In real applications, errors of the input poses occur in errors of orientation and length of the translation vector, and errors in rotation orientation. We want to analyze the effects of orientation errors of the input poses on the estimation methods. Therefore errors are emulated by the following scheme:

For orientation error analysis an *orientation error level* $\varepsilon > 0$ is given which defines the amount of rotation and position orientation error of the poses. Rotations R_i^k and positions C_i^k of each camera at each step of time are disturbed by a random rotation with an amount of ε:

$$\hat{\mathrm{R}}_i^k = \mathrm{R}_{\mathrm{r}_i^k,\varepsilon}\mathrm{R}_i^k$$

and

$$\hat{\mathrm{C}}_i^k = \mathrm{R}_{\mathrm{c}_i^k,\varepsilon}\mathrm{C}_i^k$$

where $R_{\mathrm{r},\varepsilon}$ defines a rotation around axis r by angle ε and r_i^k, c_i^k are uniform randomly chosen rotation axes.
The orientation error level ε will be specified in degree in all studies.

We observe the average of the resulting errors for 10^5 random samples consisting of $K = 4$ corresponding poses each with an error level of ε ranging from 0° to 3° in steps of 0.1°. From the results we can deduce the stability of both calibration method against noise on the orientations of the input poses. In fig. 5.1 the results for the linear least squares approach are shown. Figure 5.2 shows the average estimation error for additional non-linear refinement of the LLS results.

Apparently, for frame-independent, unbiased orientation errors on the input poses the error of the results is in the same order of size as the input noise and evolves linearly with the input error. As expected from the theoretical design, the decoupled estimation method (i.e. the linear least squares estimation) yields a greater estimation error for the internal positions because the rotation estimation errors are transferred to position estimation. The combined non-linear estimation method yields better results for position and scale estimation while rotation estimation is not significantly improved.

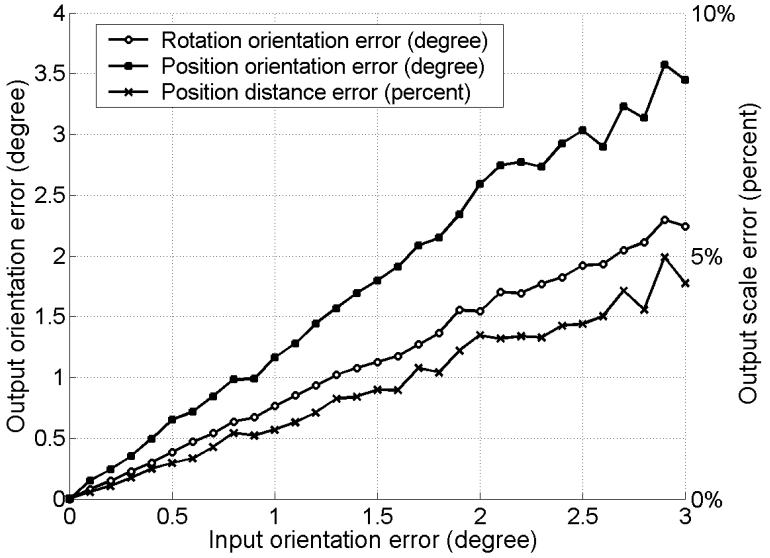


Figure 5.1.: Output error for linear least squares estimation from synthetically generated corresponding poses for 2 cameras and 4 images, depending on orientation error of the input poses. The output error is measured as (a) rotation orientation error between estimated rotation $\Delta\hat{\mathrm{R}}_i$ and real rotation $\Delta\mathrm{R}_i$, (b) position orientation error between estimated position $\Delta\hat{\mathrm{C}}_i$ and ground truth position $\Delta\mathrm{C}_i$, and (c) distance error of estimated internal position $\|\Delta\hat{\mathrm{C}}_i\|$ in percent of $\|\Delta\mathrm{C}_i\|$.

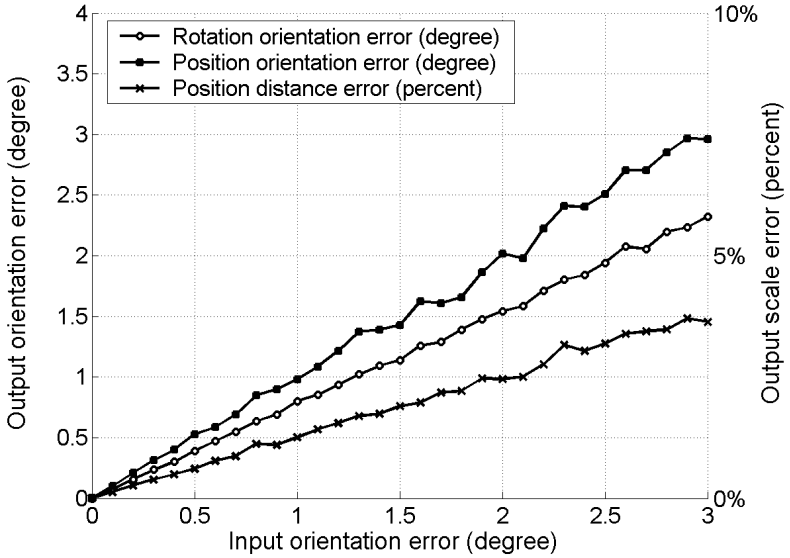


Figure 5.2.: Output error for non-linear refinement of linear least squares estimates from synthetically generated corresponding poses for 2 cameras and 4 images depending on orientation error of the input poses. The initial solution is computed by the linear least squares method. The output error is measured as in fig. 5.1.

5.1.3. Stability Depending on Number of Poses

In the previous section the performance of the proposed estimation methods was investigated for fixed a number of $K = 4$ input pose correspondences. Although a solution for internal rotation and position can be computed from a minimal set with $K = 2$ pose correspondences without noise, in the presence of measurement errors a larger set is favorable to suppress noise. In this section it will be shown how the performance improves in the test case described above when a larger number of input poses is used.

Test data is generated the same way as described in the previous section. Here the error level is fixed at $\varepsilon = 1^\circ$ and the number of corresponding input poses from which the rig parameters are estimated increases from $K = 4$ to $K = 24$. The errors of the estimation results is estimated as the average from 10^5 random samples for each $K = 4, \ldots, 24$ and plotted in fig. 5.3 for the linear least squares method and in fig. 5.4 for additional non-linear refinement. For both methods the average rotation and position orientation error is significantly decreasing with increasing number of poses used for estimation. For K greater than about 20 there is no significant improvement notable for the estimation results.

We have shown that the rig calibration error decreases with increasing number of input pose correspondences for both the linear and the non-linear calibration method. For sufficiently large numbers of pose correspondences, the error of the linear least squares method approaches zero, assuming that the input error is Gaussian distributed. Hence for large input sets, the linear approach is supposed to deliver results of equal accuracy as the non-linear approach. Nevertheless, the non-linear method is preferable for practical applications, in particular if fewer corresponding poses are available.

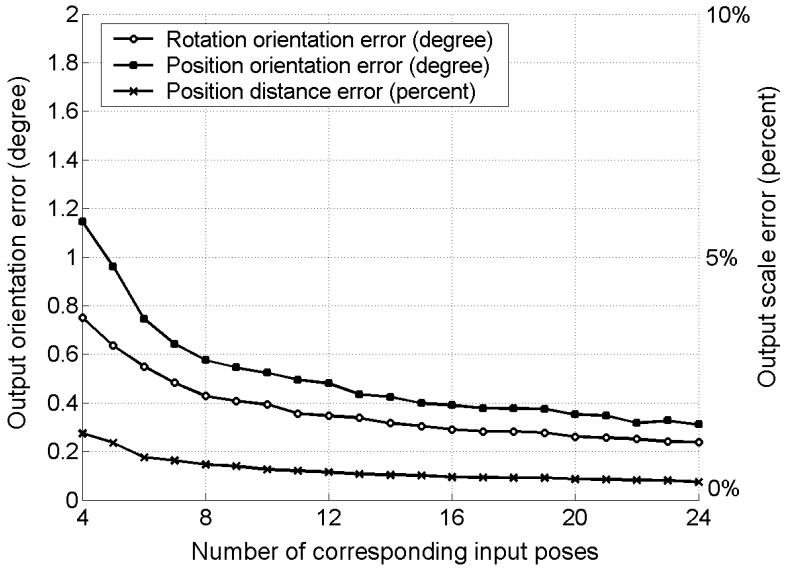


Figure 5.3.: Output error for linear least squares estimation for a stereo rig from synthetically generated corresponding poses with fixed error level ($\varepsilon = 1°$) depending on the number of input poses ranging from 4 to 24. The output error is measured as in fig. 5.1.

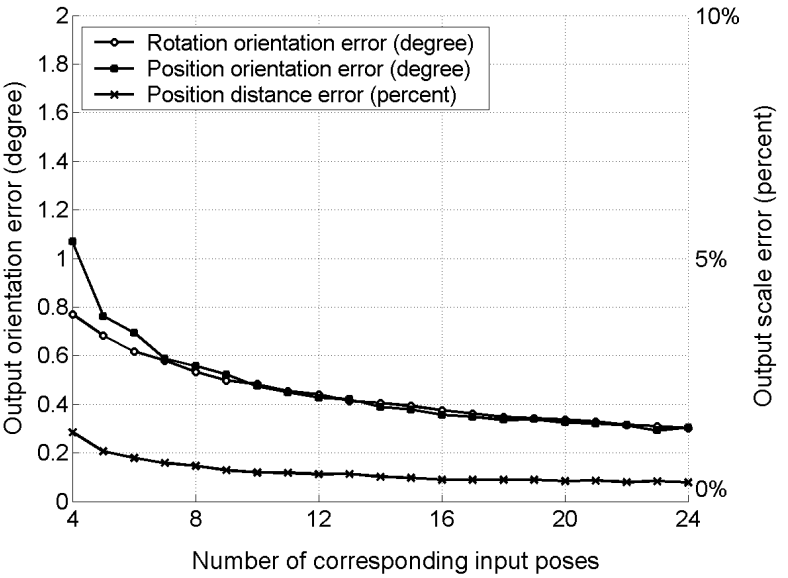


Figure 5.4.: Output error for non-linear refinement of linear least squares estimates from synthetically generated corresponding poses with fixed error level ($\varepsilon = 1°$) depending on the number of input poses. The initial solution is generated by the linear least squares method. The output error is measured as in fig. 5.1.

5.2. Comparison of General Motion Model and Purely Translational Motion Model

In order to compare estimation performance for the purely translational motion model (PTM) and general motion model as defined in section 4.2.1, both approaches are tested for a large set of synthetically generated corresponding poses as in the previous test case. Poses are generated with an rotation and postition orientation error of ε, ranging from 0° to 3° in steps of 1°. To model mostly translational motion, the maximal rotation α_{max} of the input poses is fixed, ranging from 0° to 25° in steps of 1°. For each pose estimation $K = 4$ input pose pairs were used. Each k-th pose has a rotation amount of $\frac{k}{K}\alpha_{max}$ degree around a randomly chosen axis. The linear least squares solution of both the general motion and purely translational motion approaches were computed for 10^5 random problem instance for each α_{max} and ε. Figure 5.5 compares the errors of both motion models and illustrates the equal error boundary for both algorithms.
The orientation errors of the estimated internal rotation are shown in fig. 5.5(top) with respect to input orientation error ε and maximal rig rotation α_{max}. One notices that the general motion model fails if orientation noise on the input poses is sufficiently large with respect to absolute rig rotation.
Figure 5.5(bottom) shows a surface plot of the estimation errors for both models depending on input accuracy and maximal absolute rotation. The line of equal error for both algorithms indicates a method for model selection. In conditions above the equal error boundary, the general motion model yields more accurate results than the PTM. Expecting an orientation error of about 1° on input poses we will fix a maximal absolute rig orientation of $\alpha_{maxrot} = 10^\circ$ to consider a rig as purely translational.

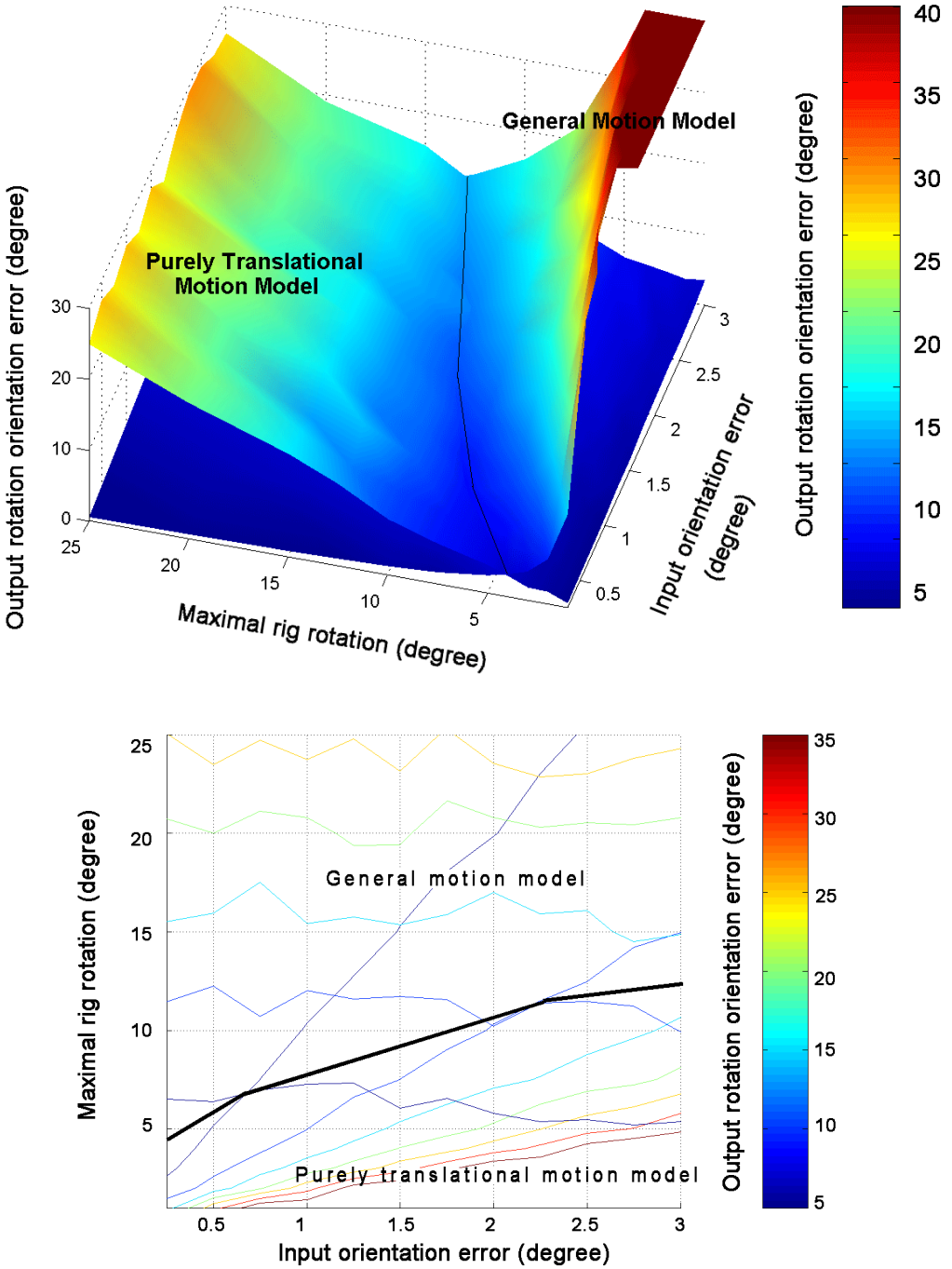


Figure 5.5.: Comparison of linear least squares estimation with general motion model and purely translational motion model. Top: Output orientation error for both motion models depending on rig rotation amount and input orientation accuracy. Bottom: Equal error boundary for general motion model (lower error above boundary) and purely translational model (lower error below boundary).

5.3. Comparison of Time-Fixed Scale Model and Time-Dependent Scale Model

In the previous sections we assumed that the error on the input data is constant over time. Nevertheless, in practical applications where the input poses are obtained by pose estimation techniques as described in sec. 3.5, the error is often observed to be biased such that the scale between camera coordinate frame at time step k and reference frame at time step 0 is varying significantly. To consider general time-dependent internal scales we defined the time-dependent scale model (TDS) in contrast to the time-fixed scale model in section $4.2.2$. In order to evaluate both approaches a similar test as for motion model comparison was prepared. Synthetically generated input poses were distorted both by an isotropic scaling and by adding an orientation error ε ranging from $0°$ to $3°$ in steps of $1°$. The maximal scale error $\varepsilon_{maxscale}$ with respect to the first pose was chosen to increase from 0% to 100% in steps of 5%. For each pose estimation $K = 4$ input pose pairs were used. The scale error for each k-th position was set to $\frac{k}{K}\varepsilon_{maxscale}$ percent. The linear least squares solutions of both the time-fixed scale and time-depending scale approaches were computed for 10^5 random problem instances for each $\varepsilon_{maxscale}$ and ε. Figure 5.6 compares the errors of both scale models and illustrates the equal error boundary for both algorithms.
The orientation errors of the estimated internal position are shown in fig. 5.6(top) with respect to input orientation error ε and maximal scale error $\varepsilon_{maxscale}$. The time-fixed scale model fails for sufficiently large scale deviations of the input positions.
Figure 5.6(bottom) shows a surface plot with marked equal error boundary. In conditions above the equal error boundary, the time-dependent scale model yields more accurate results than the time-fixed scale model and should be preferred. To assess internal scale deviation for model selection the scale measurements computed from input poses as described in section 4.3.3 can be analyzed.

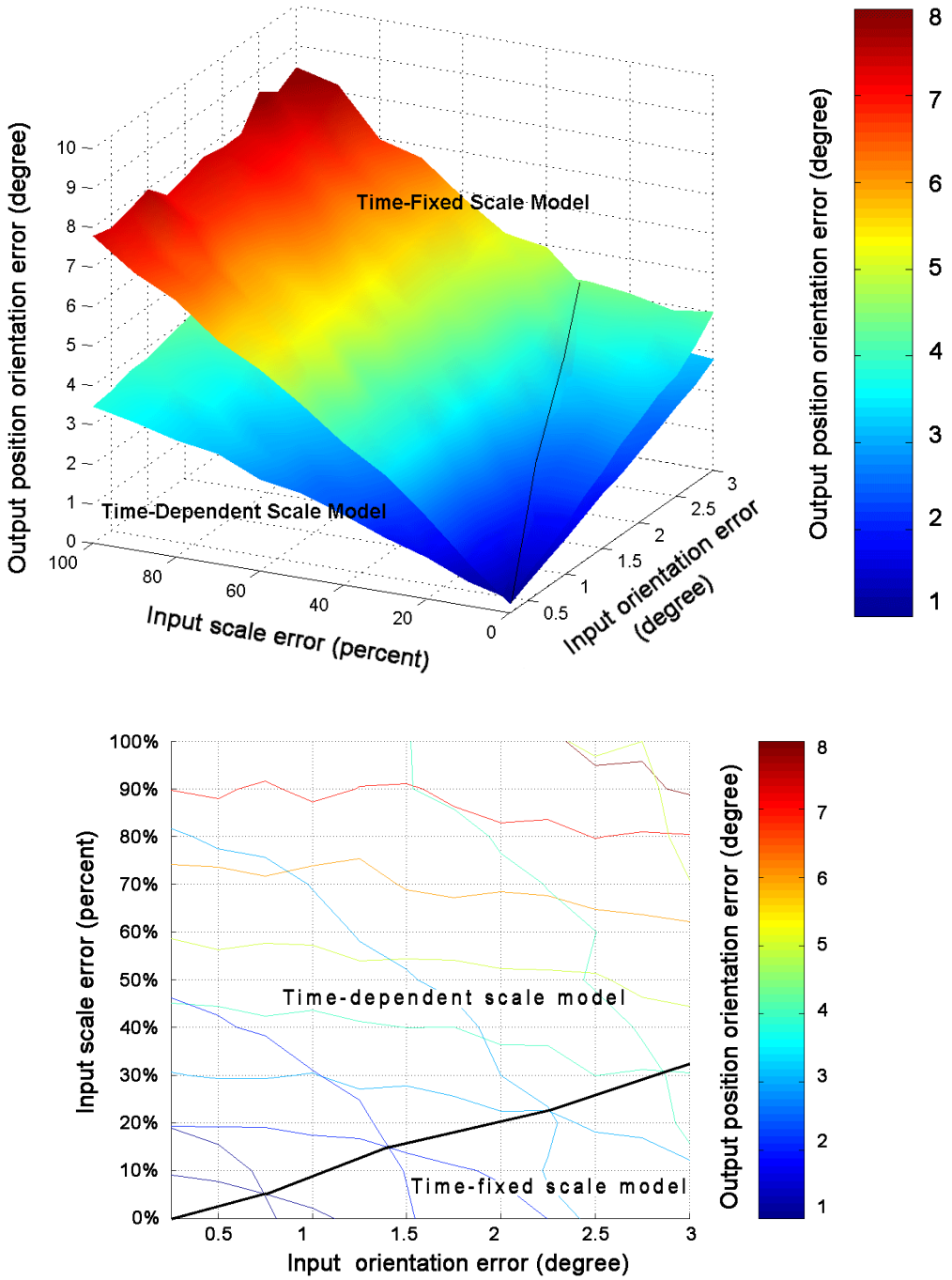


Figure 5.6.: Comparison of linear least squares estimation with time-fixed scale model and time-dependent scale model. Top: Orientation error of estimated position for both scale models depending on time-dependent error on input scales and input orientation accuracy. Bottom: Equal error boundary for time-dependent scale model (lower error above boundary) and time-fixed scale model (lower error below boundary).

5.4. Virtual Scene with Perspective Camera Rig

In the previous test synthetically generated poses were used to evaluate the stability of our approach. In the following we will apply the calibration method to poses computed from input images by pose estimation techniques as described in section 3.5. We start with an application using rendered images of a virtual scene. We use a VRML model of a scene with an OpenGL based rendering framework for image creation.

The performance of our approach should be tested with input poses that are computed from synthetic image input sequences using the pose estimation technique discussed in section 3.5. The estimated rig parameters will be compared against available ground truth data, i.e. information about the true camera poses for each image. This test case gives a good approximation of the performance of the rig calibration technique and provides absolute information about input poses and rig parameters against which the solutions from pose estimation and rig parameter estimation can be evaluated.

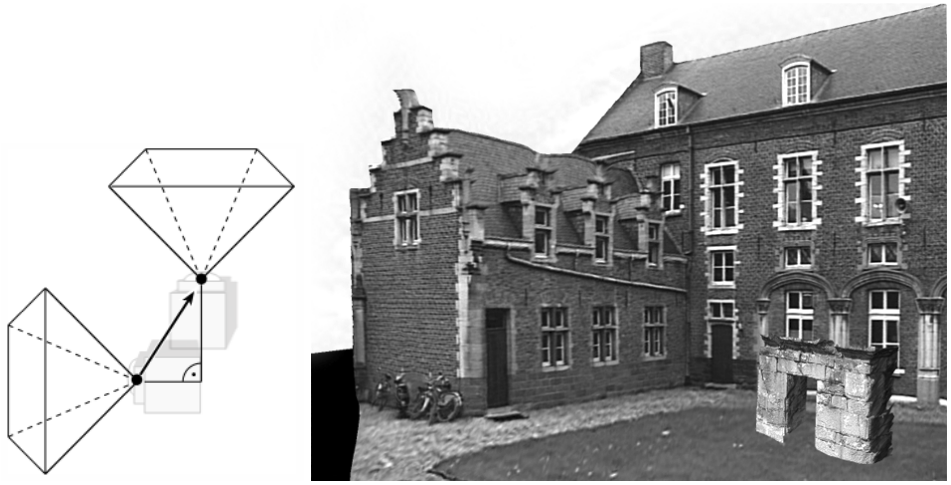


Figure 5.7.: Virtual scene application (left: rig model, right: image of virtual scene model).

We use a virtual 3d scene with an architectural model and create synthetic image sequences from a moving rig by rendering techniques. An image of the scene is shown in fig. 5.7(right). The rig model used consists of 2 perspective cameras each with an aperture angle of 60° as shown in fig. 5.7(left). The internal rig poses are given by internal position $\Delta \mathrm{C}_1 = (-1, 1, 3)^\mathsf{T}$ and internal orientation $\Delta \mathbf{q}_1 = (\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}, 0)$, i.e. the slave camera is rotated around the master camera's y-axis by an amount of 90°. The length of the virtual rig joint, the *internal rig distance*, is given by $\|\Delta \mathrm{C}_1\| \approx 3.3166$. A general motion of the rig through the scene is performed and sequences consisting of $K = 468$ images are captured synchronously for each camera hence we have time-corresponding images $\mathcal{J}_0^{0,\ldots,K}$ and $\mathcal{J}_1^{0,\ldots,K}$. The motion trails of both cameras

through the virtual scene are visualized in fig. 5.8. For each image $\mathcal{J}_i^k$ the pose of camera i with respect to world coordinate system is known. It will be denoted as $\mathbf{T}^k_{i\,world} = [\mathrm{R}^k_{i\,world}|\mathrm{C}^k_{i\,world}]$ in the following. From camera poses in the world coordinate system, ground truth poses in the camera-local coordinate frame with respect to the initial position $\mathbf{T}^0_{i\,world}$ can be computed as $\mathbf{T}^k_i = (\mathbf{T}^0_{i\,world})^{-1}\mathbf{T}^k_{i\,world}$. This was illustrated by the global reference pose model in section 4.2.

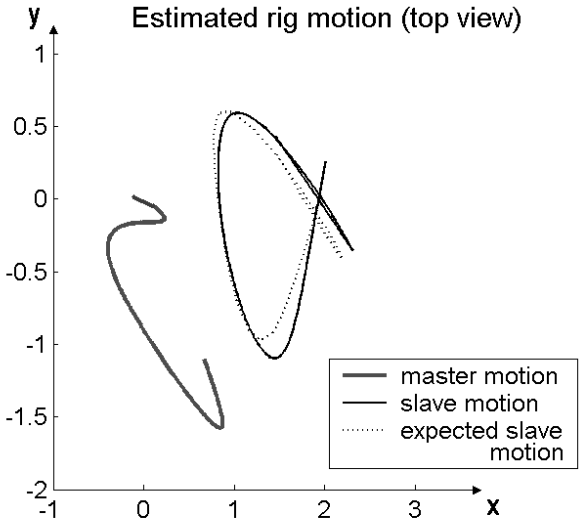


Figure 5.8.: Top-down view of a 3d plot of measured master camera motion (left solid line), expected slave camera motion (right dashed line) and measured slave camera motion (right solid line) within the master camera reference frame.

The pose estimation method described in section 3.5 is first applied to each image sequence individually. The estimated poses of each camera $i \in \{0, 1\}$ in each image $\mathcal{J}_i^k$ are denoted as $\hat{\mathbf{T}}_i^k = [\hat{\mathrm{R}}_i^k \mid \hat{\mathrm{C}}_i^k]$. The reference system for each camera is chosen such that the initial pose is $\hat{\mathbf{T}}_i^0 = [\mathrm{I} \mid 0]$. As mentioned above, the estimated poses are scaled for each reconstruction such that the baseline between the first two poses is given by $\|\mathrm{C}_i^1 - \mathrm{C}_i^0\| = 1$ resulting in the discussed scale difference $\Delta\lambda_1$ between master and slave camera reference frames. The ground truth internal scale $\Delta\lambda_1$ is therefore given by the ratio of the baseline lengths:

$$\Delta\lambda_1 = \frac{\frac{\|\mathrm{C}^1_{1\,world} - \mathrm{C}^0_{1\,world}\|}{\|\mathrm{C}^1_1 - \mathrm{C}^0_1\|}}{\frac{\|\mathrm{C}^1_{0\,world} - \mathrm{C}^0_{0\,world}\|}{\|\mathrm{C}^1_0 - \mathrm{C}^0_0\|}} = \frac{\|\mathrm{C}^1_{1\,world} - \mathrm{C}^0_{1\,world}\|}{\|\mathrm{C}^1_{0\,world} - \mathrm{C}^0_{0\,world}\|} \tag{5.3}$$

Note also that the scale between the world coordinate frame and the reconstructed master coordinate frame is given by $\Delta\lambda_{world} = \frac{\|\mathrm{C}^1_0 - \mathrm{C}^0_0\|}{\|\mathrm{C}^1_{0\,world} - \mathrm{C}^0_{0\,world}\|}$, i.e. $\Delta\lambda_{world} = 1/\|\mathrm{C}^1_{0\,world} - \mathrm{C}^0_{0\,world}\|$.

First, we analyze the quality of the pose estimation results regarding the error model

introduced in section 5.1.1. In fig. 5.9 and fig. 5.10, the rotation orientation errors (left column), position orientation errors (middle column), and distance errors (right column) of individual pose estimation for each camera are plotted for the image sequence. Figure 5.9 shows the errors for the master camera while fig. 5.10 shows the slave camera estimation errors. The upper row of each figure shows the pose error between subsequent frames which is mostly constant throughout the sequences. The lower row shows the accumulated pose estimation errors with respect to the reference frame. It appears that although errors between subsequent frames are small, the total pose errors are increasing over time due to error accumulation. The scale of the reconstruction with respect to the reference frame of each camera is also mainly increasing over time.

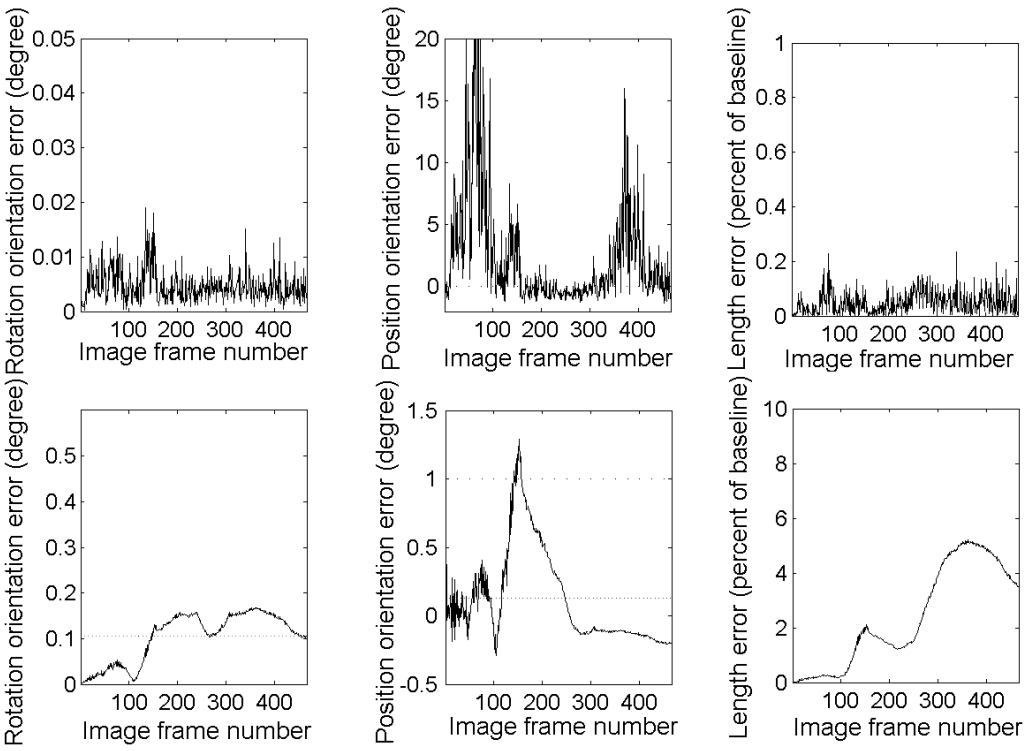


Figure 5.9.: Error analysis of pose estimation results for the master camera. The upper row shows rotation orientation error (left), position orientation error (center), and scale error (right) between subsequent image frames. The lower row shows the accumulated errors with respect to the initial image frame.

Furthermore, the impact of individual pose estimation errors on the rig constraints described in the main chapter should be investigated. Figure 5.11 shows the residua of the rotation, position, and scale constraints of the rig with respect to the corresponding estimated poses for each image frame k. In the left column, the rotation constraint residuum is shown, i.e. the orientation error between expected and measured slave pose with respect to the measured master pose and ground truth rig parameters. The

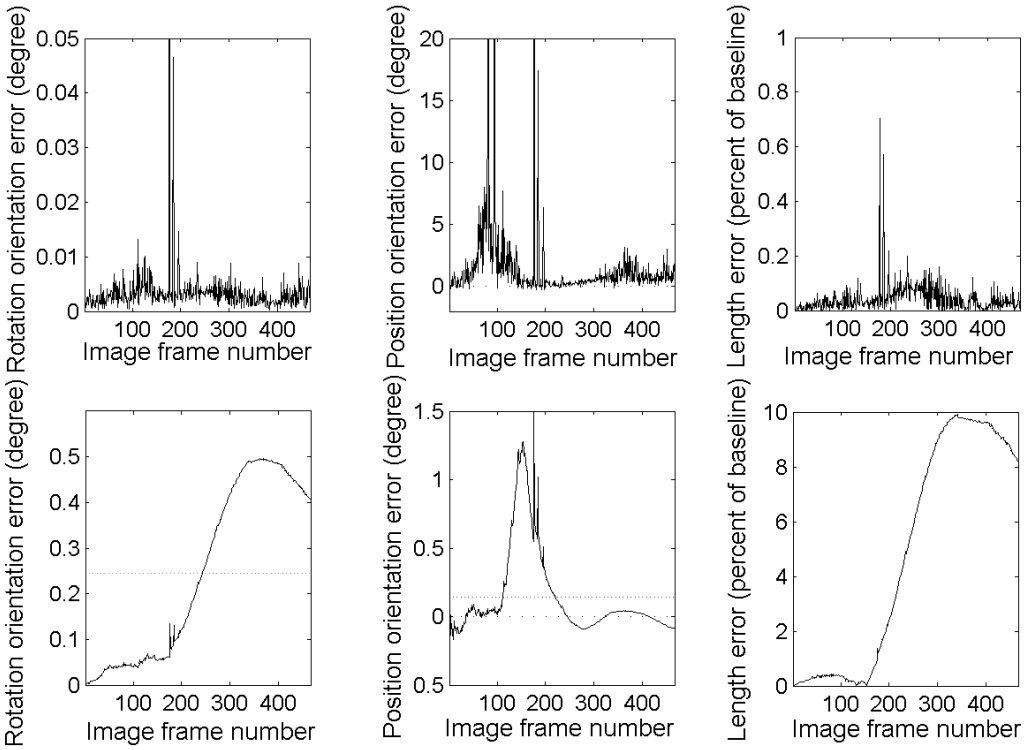


Figure 5.10.: Error analysis of pose estimation results for the slave camera. The upper row shows rotation orientation error (left), position orientation error (center), and scale error (right) between subsequent image frames. The lower row shows the accumulated errors w.r.t. the initial image frame.

middle column shows the position orientation constraint residuum, i.e. the orientation error between expected and measured slave position. The residua of both constraints are increasing significantly over time due to error accumulation of the respective corresponding poses. Note that the rig constraint residua correlate with the errors of the respective individual input poses.

To analyze how the lengths of estimated position vectors of each camera deviate with respect to each other over time, the ratios between measured master and slave position length are compared to the ground truth translation ratios for each k. The absolute difference between both ratios is shown in the right column of fig. 5.11 as:

$$\left| \frac{\frac{\|\mathrm{C}^k_{1_{world}} - \mathrm{C}^0_{1_{world}}\|}{\|\mathrm{C}^k_1 - \mathrm{C}^0_1\|}}{\frac{\|\mathrm{C}^k_{0_{world}} - \mathrm{C}^0_{0_{world}}\|}{\|\mathrm{C}^k_0 - \mathrm{C}^0_0\|}} - \Delta\lambda_1 \right| / \Delta\lambda_1 \tag{5.4}$$

For the first time step $k = 1$, the measured ratio equals the ground truth ratio by definition but it diverges over time.

At last, the scale measurements λ^k_i directly computed from the estimated correspond-

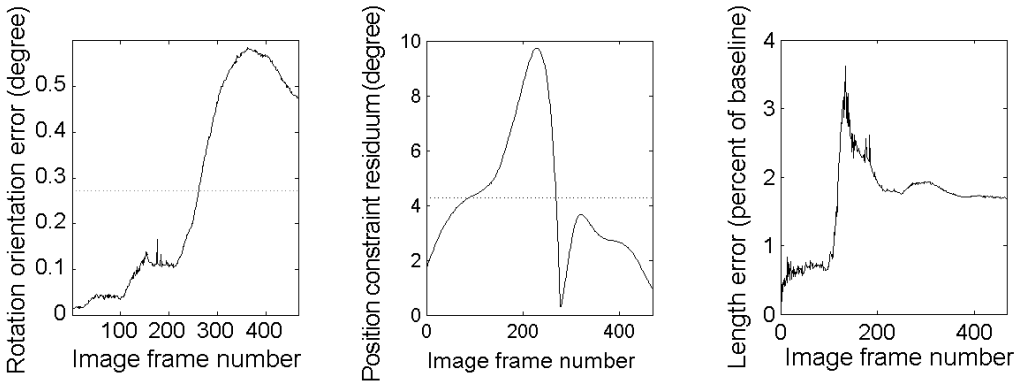


Figure 5.11.: Error analysis of rig constraints between estimated master and slave poses with respect to ground truth rig parameters. Shows the rotation constraint residuum as orientation (left), the position constraint residuum as orientation (center) and the difference between corresponding position vector lengths (right) for each image frame.

ing poses as described in section 4.3.3 are investigated: Figure 5.12(left) shows the measured scales for each image related to the ground truth internal scale $\Delta\lambda_1$. As expected, the measured scales degenerate at certain points of time, i.e. where rig translation is mostly perpendicular to the rotation axis. In section $4.3.3$ we suggested to invalidate measured scales for motions where ${\mathrm{C}_1^k}^\mathsf{T}\mathrm{r}_1^k$ is below a determined threshold $\delta_{threshold}$. By choosing $\delta_{threshold} = 3 \cdot 10^{-3}$ as motivated in eq. (4.23) the most degenerate motions, marked with a dashed line in fig. 5.12(left), are rejected. Figure 5.12(right) shows the respective scalar products ${\mathrm{C}_0^k}^\mathsf{T}\mathrm{r}_0^k$ and ${\mathrm{C}_1^k}^\mathsf{T}\mathrm{r}_1^k$ for both cameras at each time step. The validity threshold is indicated by the dashed line.

The calibration results are compared with ground truth data in tab. 5.1. Using all corresponding poses for rig-calibration according to the general motion model and time-fixed scale model we get an internal rotation of $89.8253°$ around axis $(-0.0017, 0.9999, 0.0038)^\mathsf{T}$ and internal position vector $(-0.994, 1.028, 3.048)^\mathsf{T}$ for the linear least squares approach, and an internal rotation of $89.8703°$ around axis $(-0.0014, 0.9999, 0.0033)^\mathsf{T}$ and internal position vector $(-0.987, 1.019, 3.021)^\mathsf{T}$ for the non-linear approach using the LLS result as start value. The rotation orientation error with respect to ground truth internal rotation is given by $0.3803°$ for the LLS solution (LLS) and $0.3171°$ for the non-linear solution (NL). The position orientation error is $0.4297°$ (LLS) and $0.3889°$ (NL) while the distance error is 1.5269% (LLS) and 0.6488% (NL) as percentage of the ground truth distance. Another way to access the accuracy of our calibration results is to consider the residual errors associated with the non-linear error functional ϕ^2_{NL} from eq. (4.26). The total residual error of solution $\Delta\hat{\mathrm{R}}_1$, $\Delta\hat{\mathrm{C}}_1$, $\Delta\hat{\lambda}_1$ for all K image frames is computed to be $\phi^2_{NL}(\Delta\hat{\mathrm{R}}_1, \Delta\hat{\mathrm{C}}_1, \Delta\hat{\lambda}_1) = 0.0975595$ which is quite low.

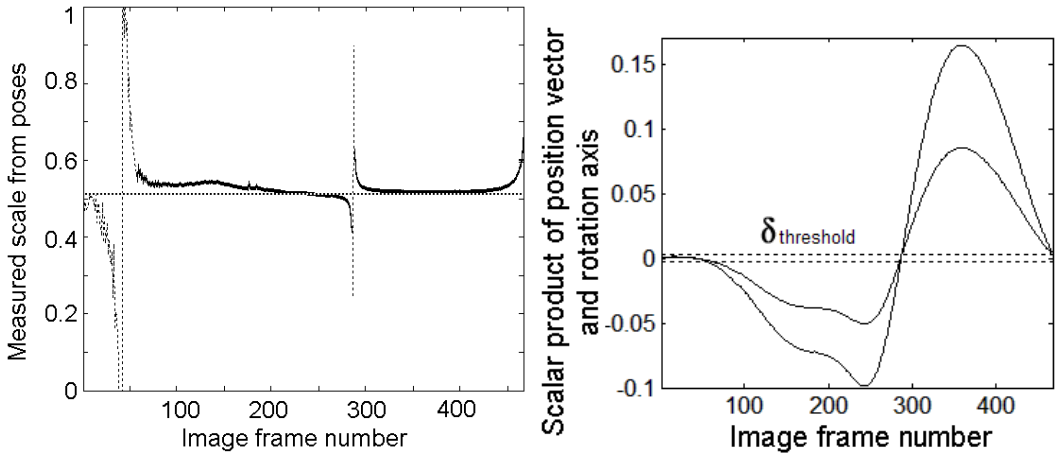


Figure 5.12.: Analysis of measured scales computed from corresponding estimated master and slave poses. The left image shows the scales for each image frame, i.e. the ratio between scalar products of rotation axis and position vector of master and slave camera. The right image shows the respective scalar products for each camera. Scales computed from scalar products below $\delta_{threshold}$ are rejected as indicated by dashed lines. The horizontal line denotes the average of valid values.

	$\Delta\mathbf{q}_1$	$\Delta\mathrm{C}_1$	$\|\Delta\mathrm{C}_1\|$
GT	$(0.7071, 0, 0.7071, 0)$	$(-1, 1, 3)$	3.3166
LLS	$(-0.706, 0.001, -0.708, -0.003)$	$(-0.994, 1.028, 3.048)$	3.3673
NL	$(-0.706, 0.001, -0.707, -0.002)$	$(-0.987, 1.019, 3.021)$	3.3381
LLS err.	$0.3803°$	$0.4297°$	1.5269%
NL err.	$0.3171°$	$0.3889°$	0.6488%

Table 5.1.: Comparison of calibration results using our linear least squares method (LLS) and non-linear refinement (NL) vs. ground truth data (GT).

The estimation results are in accord with the accuracy predicted by the synthetical experiments in section 5.1.2 with respect to the noticed accuracy of the input poses provided by image-based pose estimation. Note that the non-linear estimation method shows a slight advantage with respect to the linear least squares method. Concluding we notice that the results are appropriate to use the calibrated rig for structure from motion approaches [FKK04, Boe07].

5.5. Calibration of Stereo Rig from Overlapping Views

In the following test case the performance of our approach is compared against standard techniques for stereo rig calibration from overlapping views. We use a stereo rig as shown in fig. 5.13. The physical setup consists of two CCD cameras mounted onto a movable stand at a distance of approximately 15 cm with a relative yaw angle of about $15°$.

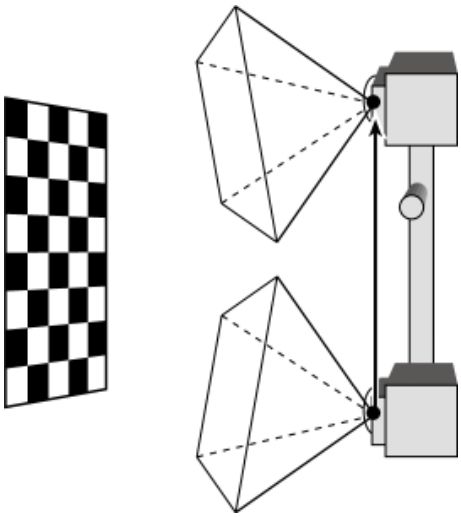


Figure 5.13.: Illustration of stereo rig calibration. The stereo rig model consists of two perspective cameras with overlapping fields of view.

The rig calibration is computed using (i) *Bouguet's* camera calibration toolbox from [Bou07] and (ii) by our approach. A description of (i) is given and both calibration results will be presented in the following.

First the rig is calibrated using *Bouguet's* camera calibration toolbox from $K^* = 24$ image pairs showing a calibration pattern with known geometry from different positions. This approach is similar to the intrinsic camera calibration approach described in section 3.4.1. The corners of the calibration pattern are extracted from the images using *Harris'* corner detector [HS88] and identified with corresponding 3d points within the same calibration reference frame $\mathcal{C}^{cal}$. Both the intrinsic camera parameters and corresponding external pose transformations $\mathbf{T}_i^{*k}$ for each image frame $k = 1, \ldots, K^*$ of each camera $i \in \{0, 1\}$ with respect to $\mathcal{C}^{cal}$ are computed by a linear least squares approach and refined using non-linear optimization as described in 3.4.1. The intermediate rig transformation $\Delta\mathbf{T}_1^*$ between master and slave camera is then computed as the average from $(\mathbf{T}_0^{*k})^{-1}\mathbf{T}_1^{*k}$ over all frames k as motivated for the global reference pose model in section 4.2. Figure 5.14(left) shows different images of master and slave

camera during the rig motion performed for calibration.

Figure 5.14.: Images from a sequence used for stereo rig calibration. The left column shows master and slave camera images for *Bouguet's* calibration using a calibration pattern. The right column shows master and slave camera images from a motion of the rig through a scene used for our approach.

Second, the rig parameters are estimated using the suggested rig calibration algorithm without utilizing special calibration patterns. We use the non-linear estimation method for the general motion model. The initial solution is provides from the linear least squares approach for the general motion model. Time-corresponding poses are estimated for an image sequence consisting of also $K = 250$ images by the feature point based pose estimation technique proposed in section 3.5. Figure 5.14(right) shows two images of master and slave camera during the rig motion performed for calibration. During the sequence, the rig is translated and rotated significantly with respect to the scene. All poses are given with respect to the respective initial camera reference frame as motivated for the camera-local pose model in section 4.2. To relate the estimated internal position to world metrics, an object of known size, i.e. a planar checkerboard pattern with 4.18×4.18 cm tiles, is visible in some images of the master camera. The scale of the master camera reference system is computed by relating the average distance between the reconstructed 3d points of adjacent checkerboard pattern corners to 4.18 cm.

In section 4.2.2 we motivated the use for the time-dependent scale model when the scale between corresponding estimated poses diverges largely over time. In order to

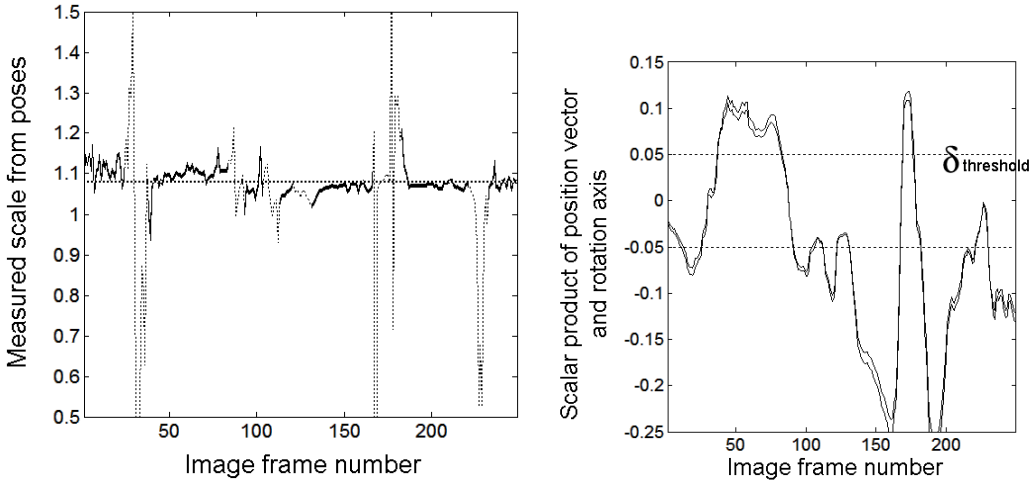


Figure 5.15.: Analysis of measured scales computed from corresponding estimated master and slave poses. The right image shows scalar products of rotation axis and translation vector for each image frame (here: approximately identical for master and slave poses). The left image shows measure scales, i.e. the ratio between the scalar products for each image frame (approximately 1). Scales computed from scalar products below $\delta_{threshold}$ are rejected. Rejected values are indicated by dashed lines in the right plot. The horizontal line denotes the average of valid values.

select the appropriate scale model, the scales computed from corresponding poses for each image frame as defined in eq. (4.18) are analyzed. In fig. 5.15(left) the computed scales are plotted using a threshold of $\delta_{threshold} = 5 \cdot 10^{-3}$ as motivated in the previous test case. Mean and deviation of valid measured scales according to eq. (4.18) are computed as 1.0813 ± 0.0295. Since the scale deviation is below 3% we can assume that the time-fixed scale model holds for this application.

	$\Delta\mathbf{q}_1$	$\Delta\mathrm{C}_1$	$\|\Delta\mathrm{C}_1\|$
ref.	$(0.993, -0.054 - 0.098 - 0.012)$	$(15.303, 1.048, -2.19)$	15.494
LLS	$(0.993, -0.053, -0.096, -0.016)$	$(15.13, 0.506, -1.77)$	15.242
NL	$(0.994, -0.049, -0.093, -0.015)$	$(15.158, 1.325, -2.371)$	15.29
LLS diff.	$0.5026°$	$2.4652°$	1.6281%
NL diff.	$0.8846°$	$1.2952°$	0.6086%

Table 5.2.: Comparison of calibration results for stereo rig from overlapping views using *Bouguet's* calibration toolbox as reference (ref.) and our approach (LLS: linear least squares method, NL: non-linear method). Result differences are given with respect to the reference calibration.

The calibration results of both approaches are compared in tab. 5.2. From *Bouguet's* approach we get an internal rotation of $12.9483° \pm 0.23°$ around axis $(-0.477, -0.872,$

$-0.107)^\mathsf{T}$ and an internal position vector $(15.303, 1.048, -2.19)^\mathsf{T} \pm (0.1940, 0.3086, 0.0654)^\mathsf{T}$ giving a distance of 15.4942 ± 0.2 cm between the centers of the rig cameras.

The linear least squares solution yields an internal rotation of $12.7644°$ around axis $(-0.484, -0.863, -0.142)^\mathsf{T}$ and internal position $(15.13, 0.506, -1.77)^\mathsf{T}$ with length 15.2419 cm. After additional non-linear refinement we have an internal rotation of $12.2175°$ around axis $(-0.466, -0.873, -0.145)^\mathsf{T}$ and internal position $(15.158, 1.325, -2.371)^\mathsf{T}$ with length 15.29 cm. One notices that the non-linear solution has an advantage in estimation the internal position vector while internal rotation is estimated more accordingly to the reference calibration by the linear least squares method. We remind that the LLS method decouples rotation estimation from position estimation. Hence position measurement errors do not affect internal rotation estimation which may results in more accurate rotation estimation for a relatively small number of input poses.

The orientation difference between the results from (i) and (ii) is estimated to be approximately $0.9°$. The direction difference of the two resulting translation vectors is estimated to be approximately $1.3°$ orientation and 0.2 cm length. Apparently the rig calibration results from (ii) are in the same order of magnitude as the results from the marker based approach (i). The total residual error of the solution of our approach for all $K = 250$ image frames is computed to be $\phi^2_{NL}(\Delta\hat{\mathrm{R}}_1, \Delta\hat{\mathrm{C}}_1, \Delta\hat{\lambda}_1) = 3.644$.

Concluding, the accuracy of our approach is within the range appropriate for general 3d reconstruction tasks using the calibrated rig according to [FKK04, Boe07].

5.6. Calibration of a Sewer Inspection System

Another test case was designed to evaluate our approach for calibrating a sewer inspection camera system. Common commercial systems such as IBAK PANORAMO®[1] depicted in fig. 5.16(right) consist of high-resolution digital cameras with wide-angle lenses attached at front and rear of a remote-controlled sledge. For theoretical evaluation a virtual scene model is used and results are compared against ground truth data as in test case 5.4.

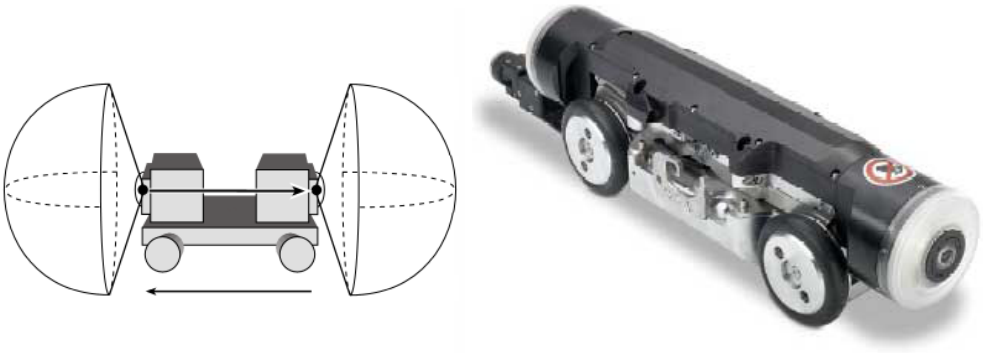


Figure 5.16.: Illustration of sewer inspection application. Left: Rig model consisting of two hemispherical cameras with non-overlapping fields of view. Right: IBAK PANARAMO® sewer inspection system consisting of two fisheye-lens cameras. By courtesy of IBAK Helmut Hunger GmbH & Co. KG.

We use a virtual rig model similar to IBAK PANORAMO®. The rig model consists of two hemispherical camera as described in section 3.2.2 each with an aperture angle of $186°$ which are rotated by $180°$ with respect to each other and translated along their optical axis by 35 cm as shown in fig. 5.16(left).

The image sequence used for calibration is created by rendering a motion of the virtual rig through a sewer pipe model with a diameter of 60 cm. The model is provided with a 2D texture created from real sewer images as shown in fig. 5.17. The motion is mainly translational along the optical axis of the rig with rig rotation below $10°$ as shown in fig. 5.18. $K = 30$ image pairs are captured during the motion of the rig each for an intermediate distance of 5 cm. To find corresponding feature points between subsequent images, the hemispherical images are first rectified by mapping them onto a virtual cylinder with a diameter of 60 cm stretched along the camera's optimal axis as shown in fig. 5.17 for a front view image. Feature points are detected and matched in the rectified images using the KLT tracking method [TK91]. This method is applicable since the geometry of the sewer pipe can be approximated by a cylinder with

[1] See IBAK website at http://www.ibak.de for further details.

Figure 5.17.: Two rendered images from the sequence for sewer inspection system calibration (left: front camera, right: rear camera, far right: Rectified front camera image).

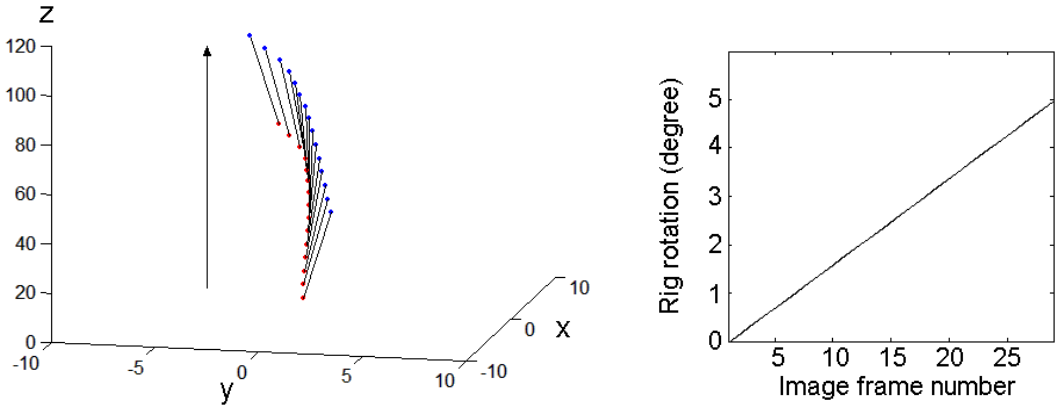


Figure 5.18.: Motion of the virtual rig during the sequence for sewer inspection system calibration (left: motion plot for front and rear camera, right: absolute rotation of front camera during rig motion).

fixed diameter and only small deviations from the translational motion of the rig along the cylinder axis are expected. Assuming an intermediate motion of 5 cm along the optical axis between subsequent images, the position of feature points can be predicted with sufficient accuracy. Hence the KLT tracker suffices a relatively small search window for feature matching. Poses of both cameras with respect to their respective initial reference frame are estimated from corresponding feature points using the pose estimation method described in section 3.5. Figure 5.19(left) and fig. 5.19(center) show the rotation orientation error of the estimated poses for front (master) and rear (slave) camera with respect to their reference frames.

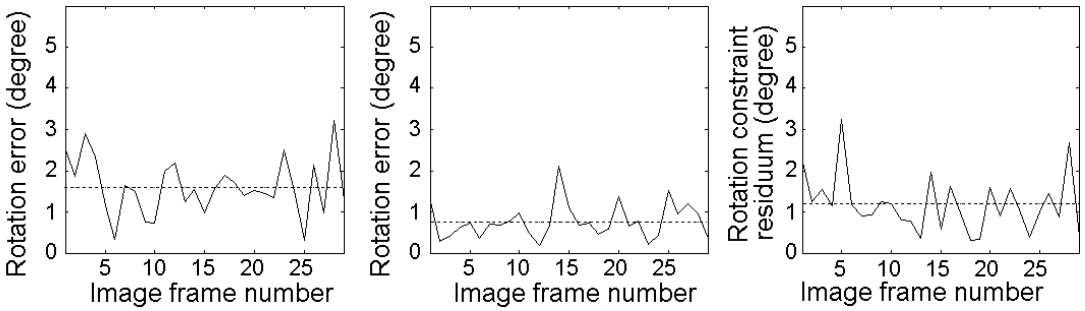


Figure 5.19.: Error analysis of pose estimation results for the sewer inspection system. The figure shows the rotation orientation error of estimated poses for front (left) and rear camera (center), and the rotation constraint error of the estimated solution (right).

Since rig rotation is expected to be minor, the purely translational motion model (PTM) is used for rig calibration. Hence we will only estimate the internal rotation $\Delta\mathrm{R}_1$ between both cameras. The solution $\Delta\hat{\mathrm{R}}_1$ is computed by the linear least squares method only. Table 5.3 shows the calibration results compared to ground truth data. Figure 5.19(right) shows the rotation constraint error of solution $\Delta\hat{\mathrm{R}}_1$, indicating the residual error of eq. (4.30), for each image frame. We notice that the rotation orientation error between estimated $\Delta\hat{\mathrm{R}}_1$ and ground truth $\Delta\mathrm{R}_1$ is computed as about $0.258°$.

	$\Delta\mathbf{q}_1$	$\Delta\mathrm{C}_1$	$\|\Delta\mathrm{C}_1\|$
GT	$(0, 0, -1, 0)$	-	-
LLS	$(-0.0003, 0.002 - 0.999, 0.001)$	-	-
LLS err.	$0.258°$	-	-

Table 5.3.: Comparison of ground truth data (GT) and calibration results using out linear least squares approach (LLS) for sewer inspection system.

We showed that internal rotation can be recovered for the synthetic sewer inspection model with a fair accuracy using the PTM approach if the rig motion is almost translational.

5.7. Calibration of a Spherical Camera

The third test case considers an implementation of a spherical camera realized by a rig of wide-angle cameras as proposed in [Koc07]. Spherical cameras, i.e. camera systems that cover a full 360° field of view, are of great interest e.g. for video surveillance applications. *Koch* proposes an implementation of a spherical camera which consists of two rigidly coupled fisheye-lens cameras each with an aperture angle of 186° which are mounted onto a pan/tilt unit. Both cameras are rotated by approximately 180° with respect to each other as shown in fig. 5.20. The translational offset between the cameras is preferably small in order to approximate a single effective projection center. Nevertheless it is bounded by the physical dimensions of the cameras. Once the internal camera poses with respect to each other are known, both hemispherical images can be assembled into an approximation of a full spherical image. This is performed by mapping both images onto a sphere with fixed radius positioned at the centroid of both image hemispheres as depicted in fig. 5.20.

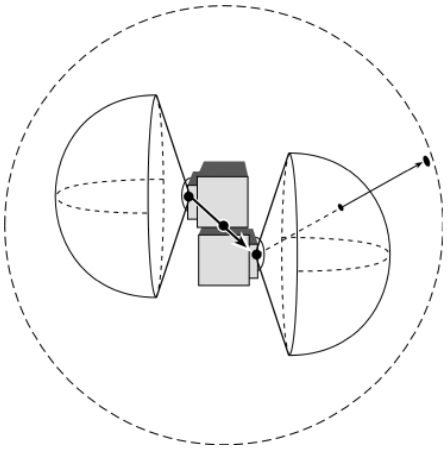


Figure 5.20.: Illustration of spherical camera application. The rig model consists of two hemispherical cameras with virtually non-overlapping fields of view mounted close to each other. The rig implements a spherical camera by mapping both hemisphere images onto a combined sphere image.

Our calibration approach for general motion and time-fixed scales is used to estimate internal position and orientation of the rig from a sequence of $K = 60$ time-corresponding images during a motion of the camera system. The motion consists of translation and rotation provided by a translation of the entire rig with concurrent pan and tilt motion of the camera supporting pan/tilt unit.

The calibration results of our approach (linear least squares and additional non-linear refinement using the general motion model and time-fixed scale model) are compared against results from an alternate calibration corresponding to a variant of the common

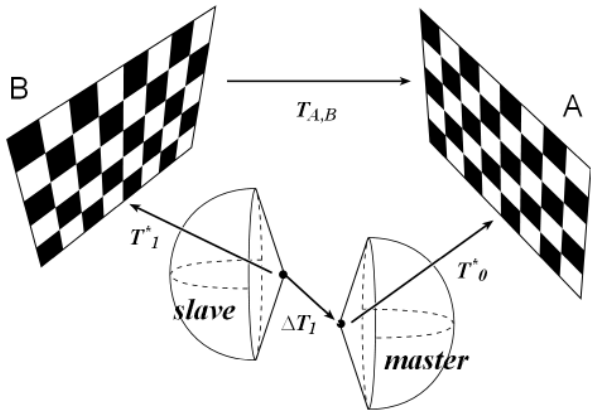


Figure 5.21.: Illustration of alternate spherical camera calibration from multiple calibration patterns. The pose transformation between both calibration patterns has to be estimated first. The respective calibration pattern must be visible in each image of the calibration sequence for each camera.

stereo-rig calibration method applied in section 5.4 using *Bouguet's* camera calibration toolbox. The general procedure of this calibration method is described in the following. We use a calibration object consisting of two planar checkerboard calibration patterns fixed at an edge such that one pattern is visible for each camera of the rig at the same time. The Euclidean transformation $\mathbf{T}_{A,B} = [\mathrm{R}_{A,B}\mathrm{C}_{A,B}]$ linking the coordinate frames $\mathcal{C}^A$, $\mathcal{C}^B$ of both calibration patterns is estimated from 4 camera images showing both patterns at the same time by estimating the homography between both patterns. From 8 time-corresponding images of the rig each showing one calibration pattern, corresponding poses $\mathbf{T}_0^{\mathcal{C}^A}$, $\mathbf{T}_1^{\mathcal{C}^B}$ of both cameras within the respective calibration coordinate frame are computed using *Bouguet's* toolbox. An estimate for the relative pose $\Delta\mathbf{T}$ of slave camera 1 with respect to master camera 0 is then computed from corresponding poses as $\Delta\mathbf{T} = \mathbf{T}_0^{\mathcal{C}^A}\mathbf{T}_{rel}(\mathbf{T}^{\mathcal{C}^B})^{-1}$. Refer to fig. 5.21 for an illustration of this calibration method.

We use this approach to calibrate the rig from 8 images resulting in an internal rotation of approximately $178.21° \pm 1.24°$ around the y-axis and an internal position vector $(-5.31, 0.6, 6.27)^\mathsf{T} \pm (-0.75, 0.32, 0.18)^\mathsf{T}$ with length 8.2446 ± 0.6645 cm.

The results of both calibration methods are compared in tab. 5.4. We observe that our results are differing more notably from the reference results than in the previous test case. Note that the reference data obtained from the proposed approach is inaccurate due to the difficult estimation of the relative positions of the calibration patterns with respect to each other. On the other hand, the resulting calibration meets the expectation by qualitative evaluation.

	$\Delta\mathbf{q}_1$	$\Delta\mathrm{C}_1$	$\|\Delta\mathrm{C}_1\|$
ref.	$(0.015, 0.144, 0.974, -0.174)$	$(-5.31, 0.6, 6.27)$	8.2446
LLS	$(0.017, 0.147, 0.977, -0.173)$	$(-4.81, 0.212, 5.32)$	7.1752
NL	$(0.017, 0.145, 0.975, -0.169)^\mathsf{T}$	$(-4.927, 0.82, 5.89)$	7.7227
LLS diff.	$0.4525°$	$3.099°$	12.91%
NL diff.	$0.6439°$	$1.95°$	6.26%

Table 5.4.: Comparison of calibration results of an approach using *Bouguet's* toolbox as reference (ref.) vs. our linear least squares method (LLS) and non-linear refinement (NL) for spherical camera rig. Result differences are given with respect to the reference calibration.

6. Conclusion

Within the scope of this work we developed and tested a novel approach for calibrating the internal poses of a multi-camera rig using time-corresponding poses provided by an image-based pose estimation method. The method was inspired by hand-eye calibration approaches and was designed inter alia for the purpose of 3d scene reconstruction using the calibrated rig. The advantage of our approach is the fact that no overlapping images are needed to calibrate the rig as most reference approaches demand. Both a linear and a non-linear method were proposed and compared.

Ill-conditioned situations, namely purely translational motion of the rig, were pointed out and an alternate method for this case limited to the estimation of internal rotation was proposed.

Providing the time-corresponding input poses by an image-based pose estimation method, we encountered the problem that the related coordinate frames of the cameras differ each by certain scale factors. Such scale factors are in general not considered in hand-eye estimation. These parameters result from the scale ambiguity that appears at the initialization of the pose estimation method and must be estimated along with the internal pose parameters. Our analysis showed that the quality of the estimated internal position depends largely on the accuracy of the estimated scales. Additional constraints on the scales were used to stabilize the estimation.

The proposed method was evaluated against established methods for stereo rig calibration and proved to be comparable in accuracy. Furthermore, we showed that it is suited for several practical applications where a calibrated rig with almost non-overlapping camera views is sufficient. However, we were confronted with the problem of obtaining reference data for the case of non-overlapping views since there are no general rig calibration methods present that do not depend on overlapping views.

Although we use a pose-estimation approach based on structure from motion to provide time-corresponding poses, our approach decouples rig calibration in general from 3d reconstruction. Hence deliberate motions can be performed for accurate rig calibration. In addition, the calibration can also be achieved in a non-overlapping setup where no marker calibration is possible. Of course the accuracy is dependent on the results of the algorithm by which time-corresponding poses are provided.

Future Work

Future work based on the proposed approach could investigate the benefit of direct integration of the calibration process into the structure from motion algorithm. Other methods circumventing the dependency of the calibration on structure from motion results could be found and investigated likewise, i.e. by adding sensors to the rig setup. Furthermore, an extension of our approach to synchronous estimation of intrinsic and extrinsic parameters could be researched.

A. Appendix I: Theoretical Topics

A.1. Singular Value Decomposition

The *singular value decomposition* (SVD) provides a factorization of a rectangular matrix A which can be applied to solve several numerical problems such as e.g. matrix inversion or eigenvalue problems. For a detailed theoretical and practical description of the SVD refer to [WR71].
The general theorem of the SVD for real matrices states that for every matrix $\mathrm{A} \in \mathbb{R}^{m \times n}$ there is a factorization - called the singular value decomposition of A - of the form:

$$\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^{\mathsf{T}} \tag{A.1}$$

where $\mathrm{U} \in \mathbb{R}^{m \times n}$ is a column-orthogonal matrix, $\mathrm{V} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, and $\Sigma \in \mathbb{R}^{n \times n}$ is a positive definite diagonal matrix of the form $\mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ consisting of the so-called *singular values* $\sigma_1, \ldots \sigma_n$ which are in fact the non-negative roots of the *eigenvalues* of $\mathrm{A}^{\mathsf{T}}\mathrm{A}$. Typically, the singular values are sorted as $\sigma_1 \geq \cdots \geq \sigma_n$ which makes Σ unique though U and V are not necessarily uniquely determined.

Computation of the pseudoinverse: Given a singular value decomposition $\mathrm{U}\Sigma\mathrm{V}^{\mathsf{T}}$ of a rectangular matrix A, the pseudoinverse of A is then represented by:

$$\mathrm{A}^{\dagger} = \mathrm{V}\Sigma^{+}\mathrm{U}^{\mathsf{T}} \tag{A.2}$$

where Σ^{+} is the transpose of Σ with every non-zero entry replaced by its reciprocal such that $\Sigma^{+}\Sigma = \Sigma\Sigma^{+} = \mathrm{I}$. Apparently for regular square matrices A the pseudoinverse $\mathrm{A}^{\dagger}$ coincides with the inverse A^{-1}.

Solution of linear least squares problems: A linear least squares problem (L) with squared error function $\phi^2(\mathrm{x}) = \|\mathrm{Ax} - \mathrm{b}\|^2$ can be solved by a SVD of A even if A has not full rank and is ill-conditioned. From the factorization $\mathrm{A} = \mathrm{U}\Sigma\mathrm{V}^\mathsf{T}$ the pseudoinverse $\mathrm{A}^\dagger = \mathrm{V}\Sigma^+\mathrm{U}^\mathsf{T}$ is obtained. Thus, the solution x^* for (L) can be calculated as:

$$\mathrm{x}^* = \mathrm{V}\Sigma^+\mathrm{U}^\mathsf{T}\mathrm{b} \tag{A.3}$$

Computation of eigenvalues: Given a singular value decomposition $\mathrm{U}\Sigma\mathrm{V}^\mathsf{T}$ of a rectangular matrix A, the eigenvalue/eigenvector decomposition of the symmetric square matrix $\mathrm{A}^\mathsf{T}\mathrm{A}$ into an orthogonal $n \times n$ matrix $\mathrm{W} = (\mathrm{w}_1 \ldots \mathrm{w}_n)$ consisting of the eigenvectors $\mathrm{w}_1, \ldots, \mathrm{w}_n$ and a diagonal $n \times n$ matrix Λ consisting of the eigenvalues can be computed by:

$$\mathrm{A}^\mathsf{T}\mathrm{A} = \mathrm{W}\Lambda\mathrm{W}^\mathsf{T} = \mathrm{V}\Sigma^\mathsf{T}\mathrm{U}^\mathsf{T}\mathrm{U}\Sigma\mathrm{V}^\mathsf{T} = \mathrm{V}\Sigma^\mathsf{T}\Sigma\mathrm{V}^\mathsf{T}$$

hence:

$$\mathrm{W} = \mathrm{V} \qquad \Lambda = \Sigma^\mathsf{T}\Sigma = \mathrm{diag}(\sqrt{\sigma_1}, \ldots, \sqrt{\sigma_n}) \tag{A.4}$$

Numerical stability of SVD: As stated above the decomposition can always be performed even if A is singular. Numerical instabilities can arise when the condition number of A - i.e. the ratio between the largest and the smallest non-zero singular value - becomes too large. This is the case when there are very small but non-zero singular values present due to noise. In the implementation of the SVD used, this problem is circumvented by *truncation*, i.e. singular values below a certain threshold are set to zero ignoring them, when the condition number of A is considered too large.

A.2. Representations of Rotation

Rotation matrices Each rotation in Euclidean 3d space is defined uniquely by an orthonormal 3×3 matrix R, i.e. a matrix for which holds $\mathrm{R}^\mathsf{T}\mathrm{R} = \mathrm{R}\mathrm{R}^\mathsf{T} = \mathrm{I}$. Rotation matrices have been the most common representation for rotations in practical applications because they can be applied fast by computing linear matrix-vector products. On the downside, rotation matrices are highly over-parametrized, comprising 9 parameters vs. 3 degrees of freedom[1]. Moreover, the orthonormality constraint is rather difficult to maintain for estimations.

Angle and axis Each 3d rotation can be described uniquely by a rotation of α degree around a rotation axis $\mathrm{r} \in \mathbb{R}^3$ with $\|\mathrm{r}\| = 1$. The angle-axis representation is formalized by a single vector $\mathrm{t} = \alpha\mathrm{r}$ such that $\mathrm{r} = \frac{\mathrm{t}}{\|\mathrm{t}\|}$ and $\alpha = \|\mathrm{t}\|$. A similar representation is the **Rodrigues vector** $\tan(\frac{\alpha}{2})\mathrm{r}$. Rotation of a vector v is performed by *Rodrigues'* rotation formula

$$\mathrm{R}_\mathrm{t}\mathrm{v} = \mathrm{R}_{\mathrm{r},\alpha}\mathrm{v} = (\mathrm{I} + \sin\alpha[\mathrm{r}]_\times + (1 - \cos\alpha)[\mathrm{r}]_\times^2)\mathrm{v} \tag{A.5}$$

where $[\mathrm{r}]_\times$ denotes the skew-symmetric cross product matrix of r. By eq. (A.5) the angle-axis representation $\alpha\mathrm{r}$ can be transformed into the rotation matrix representation given by $\mathrm{R}_{\mathrm{r},\alpha}$. Vice versa, given any rotation matrix R, the respective rotation axis r is the eigenvector of R with respect to the unit eigenvalue. It can be found by solving $(\mathrm{R} - \mathrm{I}) = \mathrm{r}$. Rotation angle α can be obtained from R by

$$2\cos\alpha = \mathrm{trace}(\mathrm{R}) - 1,\ 2\sin\alpha = (\mathrm{R}_{3,2} - \mathrm{R}_{2,3}, \mathrm{R}_{1,3} - \mathrm{R}_{3,1}, \mathrm{R}_{2,1} - \mathrm{R}_{1,2})^\mathsf{T}\mathrm{r}$$

and

$$\alpha = \arctan(\sin\alpha, \cos\alpha) \tag{A.6}$$

using a two-argument $\arctan$ function such as `atan2` in C (as pointed out in [HZ00, A4.3], applying either $\arcsin$ or $\arccos$ is numerically inaccurate and fails for $\alpha = \pi$). Note that this representation suffers from an ambiguity of the axis for $\alpha = 0$.

Euler angles Euler angles describe a 3d rotation by 3 subsequent rotations with respect to a *fixed* world coordinate system. α denotes the rotation around the fixed x-axis, β around the fixed y-axis and γ around the fixed z-axis. By convention the rotations are performed in order x-axis first, then y-axis, and z-axis last. This rotation can also be interpreted as a sequence of rotations around the *rotated* axes in the opposite order z-axis first, then y-axis, and x-axis last. The rotation matrix $\mathrm{R}_{\alpha,\beta,\gamma}$ corresponding to the Euler angles α, β, γ is composed from:

$$\mathrm{R}_{\alpha,\beta,\gamma} = \mathrm{R}_{\mathrm{b}_z,\gamma}\mathrm{R}_{\mathrm{b}_y,\beta}\mathrm{R}_{\mathrm{b}_x,\alpha} \tag{A.7}$$

[1] In fact, a general rotation in 3d space possesses 3 degrees of freedom.

where $\mathrm{b}_x, \mathrm{b}_y, \mathrm{b}_z$ is the base defining the world coordinate system. The opposite transformation yields ambiguities and is potentially instable. While the Euler angle representation provides a minimal representation with 3 parameters and is rather intuitive, it suffers from ambiguities due to the rotation order, and allows the loss of one degree of freedom under certain conditions which is referred to in literature as the *Gimbal lock* [FW04].

While rotations represented by axis r and angle α can be easily visualized, this representation is not necessary for computational purposes. The same accounts for the representation by the Euler angles. Furthermore these representations suffer from the need to evaluate trigonometric terms. Rotation matrices on the other hand can be applied easily by means of linear algebra but are highly over-parametrized.
Today the most common representation for rotations in computer vision is the *unit quaternion* [FW04]. In fact, rotations expressed by quaternions can be applied by performing linear computations with few single operations than needed for rotation matrices. Moreover, in most applications quaternions provide a both numerically and notionally far more compact representation for rotations than using rotations matrices or Euler angles. Quaternions in general and unit quaternions for rotation representation will be discussed in the following section.

A.3. Quaternions

The mathematical construct of the *quaternion* was originally developed by *William R. Hamilton* in 1866. Quaternions build an algebra $\mathbb{H}$ which can be thought of as an extension of the real algebra $\mathbb{R}$ similar to the complex algebra $\mathbb{C}$. A quaternion $\mathbf{q} = \langle q, \mathrm{q} \rangle$ consists of a scalar part q and a vector part $\mathrm{q} = (x, y, z)^\mathsf{T}$. For convenience we will identify a quaternion $\mathbf{q} = \langle 0, \mathrm{v} \rangle$ with zero scalar part simply with its vector part v.
Quaternion addition is defined componentwise. The multiplication $\mathbf{p} = \mathbf{q} \cdot \mathbf{q}'$ of two quaternions $\mathbf{q} = \langle q, \mathrm{q} \rangle$ and $\mathbf{q}' = \langle q', \mathrm{q}' \rangle$ is defined as:

$$p = qq' - \mathrm{q}^\mathsf{T}\mathrm{q}'$$

and

$$\mathrm{p} = q'\mathrm{q} + q\mathrm{q}' + \mathrm{q} \times \mathrm{q}' \tag{A.8}$$

It is a useful fact that quaternions can be equivalently expressed as 4-vectors $\mathbf{q} \equiv (q, x, y, z)^\mathsf{T}$ in terms of linear algebra. This representation is useful for direct calculations within the framework of linear algebra and is commonly used in computer vision.

Using this representation, the quaternion multiplication $\mathbf{p} = \mathbf{q} \cdot \mathbf{q}'$ can be written as a matrix-vector product $\mathbf{p} = \mathbf{T}_\mathbf{q}\mathbf{q}'$ with the 4×4-matrix $\mathbf{T}_\mathbf{q}$ induced by the 4-vector $\mathbf{q}$ (*left quaternion multiplication matrix*)

$$\mathbf{T}_\mathbf{q} = \begin{pmatrix} q & -x & -y & -z \\ x & q & -z & y \\ y & z & q & -x \\ z & -y & x & q \end{pmatrix} \tag{A.9}$$

or equivalently as a matrix-vector product $\mathbf{p} = \mathbf{T}^*_{\mathbf{q}'}\mathbf{q}$ with the 4×4-matrix $\mathbf{T}^*_{\mathbf{q}'}$ induced by the 4-vector $\mathbf{q}'$ (*right quaternion multiplication matrix*)

$$\mathbf{T}^*_{\mathbf{q}'} = \begin{pmatrix} q' & -x' & -y' & -z' \\ x' & q' & z' & -y' \\ y' & -z' & q' & x' \\ z' & y' & -x' & q' \end{pmatrix} \tag{A.10}$$

Note that quaternion multiplication is in general *not* commutative.

The inverse element of $\mathbf{q}$ with respect to multiplication is defined by:

$$\mathbf{q}^{-1} = \mathbf{q}^*/\|\mathbf{q}\|^2 \tag{A.11}$$

with the conjugated quaternion:

$$\mathbf{q}^* = \langle q, -\mathrm{q} \rangle \equiv (q, -x, -y, -z)^\mathsf{T} \tag{A.12}$$

and the quadratic norm:

$$\|\mathbf{q}\|^2 = q^2 + \mathrm{q}^\mathsf{T}\mathrm{q} \equiv \mathbf{q}^\mathsf{T}\mathbf{q} = q^2 + x^2 + y^2 + z^2 \tag{A.13}$$

Unit quaternions $\mathbf{q}$ have the norm $\|\mathbf{q}\| = 1$. It can easily be verified that the following equations hold for unit quaternions:

$$\mathbf{q}^{-1} = \mathbf{q}^*$$

and:

$$\mathbf{T}_{\mathbf{q}^{-1}} = \mathbf{T}_{\mathbf{q}}^{-1} \qquad \mathbf{T}_{\mathbf{q}^*} = \mathbf{T}_{\mathbf{q}}^{\mathsf{T}}$$

Representing rotations by unit quaternions

Unit quaternions $\mathbf{q}$ allow the representation of rotations in 3d space. Given a unit quaternion $\mathbf{q} = \langle \cos\frac{\alpha}{2}, \sin\frac{\alpha}{2} \cdot \mathrm{r} \rangle$ for a unit vector r and an angle $\alpha \in [0, 2\pi)$, the term $\mathbf{q} \cdot \mathbf{v} \cdot \mathbf{q}^*$ rotates the vector part v of $\mathbf{v}$ around axis r by an angle of α [FW04].

The rotation of a Euclidean 3d vector $\mathrm{v} \in \mathbb{R}^3$ according to $\mathbf{q}$ can hence be performed by:

$$\begin{pmatrix} 0 \\ \mathrm{R}_{\mathbf{q}}\mathrm{v} \end{pmatrix} = \mathbf{q} \cdot \mathrm{v} \cdot \mathbf{q}^* = \mathbf{T}_{\mathbf{q}}\mathbf{T}_{\mathbf{q}^*}^* \begin{pmatrix} 0 \\ \mathrm{v} \end{pmatrix} \tag{A.14}$$

where $\mathrm{R}_{\mathbf{q}} = (\mathbf{T}_{\mathbf{q}}\mathbf{T}_{\mathbf{q}^*}^*)_{(2,2)}$ denotes the *quaternion-vector rotation matrix*.

Proof: Given $\mathbf{q} = \langle \cos\frac{\alpha}{2}, \sin\frac{\alpha}{2} \cdot \mathrm{r} \rangle$ with unit vector $\mathrm{r} = (u, v, w)^{\mathsf{T}}$ and $0 \leq \alpha < 2\pi$, we receive from eq. (A.9) and eq. (A.10) and the addition theorems for sine and cosine:

$$\begin{aligned}
\begin{pmatrix} 1 & 0 \\ 0 & \mathrm{R}_{\mathbf{q}} \end{pmatrix} &= \mathbf{T}_{\mathbf{q}}\mathbf{T}_{\mathbf{q}^*}^* \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c^2 - s^2 + 2s^2u^2 & 2s^2uv - 2scw & 2s^2uw + 2scv \\ 0 & 2s^2uv + 2scw & c^2 - s^2 + 2s^2v^2 & 2s^2vw - 2scu \\ 0 & 2s^2uw - 2scv & 2s^2vw + 2scu & c^2 - s^2 + 2s^2w^2 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 + (1-c_2)u^2 & (1-c_2)uv - s_2w & (1-c_2)uw + s_2v \\ 0 & (1-c_2)uv + s_2w & c_2 + (1-c_2)v^2 & (1-c_2)vw - s_2u \\ 0 & (1-c_2)uw - s_2v & (1-c_2)vw + s_2u & c_2 + (1-c_2)w^2 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{R}_{\mathrm{r},\alpha} \end{pmatrix}
\end{aligned}$$

where $s := \sin\frac{\alpha}{2}$, $c := \cos\frac{\alpha}{2}$ and $s_2 := \sin\alpha$, $c_2 := \cos\alpha$. This provides transformations between quaternion, rotation matrix, and angle-axis representations for 3d rotations.

■

Straightforward, the concatenation of rotations corresponding to $\mathbf{q}$, $\mathbf{p}$ is given by:

$$\mathrm{R}_{\mathbf{p}\cdot\mathbf{q}} = \mathrm{R}_{\mathbf{p}}\mathrm{R}_{\mathbf{q}} \tag{A.15}$$

and the inverse rotation is given by the conjugate quaternion:

$$\mathrm{R}_{\mathbf{q}}^{-1} = \mathrm{R}_{\mathbf{q}^*} = \mathrm{R}_{\mathbf{q}}^{\mathsf{T}} \tag{A.16}$$

Note that for each unit quaternion $\mathbf{q}$, both $\mathbf{q}$ and $-\mathbf{q}$ define the same rotation. But the quaternion representation for rotation is the only one which, except for the sign ambiguity, is unique and shows no singularities [FW04].

For analytical purposes we want to give the partial derivations of quaternion multiplication and quaternion-vector rotation.

Because all quaternion multiplications can be expressed as matrix multiplications interpreting them in terms of linear algebra, their analytical derivatives can be instantly given. E.g. for left quaternion multiplication $\mathbf{p}(\mathbf{q}) = \mathbf{r}\cdot\mathbf{q} = \mathbf{T}_{\mathbf{r}}\mathbf{q}$, the partial derivations with respect to $\mathbf{q}$ are given by $\nabla_{\mathbf{q}}\mathbf{f}(\mathbf{q}) = \mathbf{T}_{\mathbf{r}}$.

Quaternion-vector multiplication as defined in eq. (A.14) can be expressed using subsequent matrix multiplications as $\mathrm{u}(\mathbf{q},\mathrm{v}) = \mathrm{R}_{\mathbf{q}}\mathrm{v} = (\mathbf{T}_{\mathbf{q}}\mathbf{T}^*_{\mathbf{q}^*})_{(2,2)}\mathrm{v} = (\mathbf{T}^*_{\mathbf{q}^*}\mathbf{T}^*_{\mathrm{v}})_{(2,1)}\mathbf{q}$. For convenience we define $\mathrm{R}'_{\mathbf{q},\mathrm{v}} = (\mathbf{T}^*_{\mathbf{q}^*}\mathbf{T}^*_{\mathrm{v}})_{(2,1)}$. Then the partial derivations with respect to $\mathbf{q}$ and v are given by $\nabla_{\mathbf{q}}\mathrm{u}(\mathbf{q},\mathrm{v}) = 2\mathrm{R}'_{\mathbf{q},\mathrm{v}}$ and $\nabla_{\mathrm{v}}\mathrm{u}(\mathbf{q},\mathrm{v}) = \mathrm{R}_{\mathbf{q}}$ respectively.

A.4. Closed-Form Solution for Absolute Orientation

In this section we will describe the closed-form solution for estimating absolute orientation using unit quaternions as proposed by *Horn* [Hor87] for finding the best rotation between two sets of 3d features. We will only need the simple case that both feature sets are located at the unit sphere.

Given are unit length vectors, or "rays", $\mathrm{r}_k = (x_k, y_k, z_k)^\mathsf{T}$, $\mathrm{r}'_k = (x'_k, y'_k, z'_k)^\mathsf{T}$ for $k = 1, \ldots, K$ where the ray set $\mathrm{r}'_1, ..., \mathrm{r}'_K$ arises from $r_1, \ldots, r_K$ by a certain rotation and Gaussian noise addition.

We now search for a rotation R which describes the transformation of the ray set $\mathrm{r}_1, \ldots, \mathrm{r}_K$ into $\mathrm{r}'_1, \ldots, \mathrm{r}'_K$ the best, i.e. which minimizes the squared error function:

$$\phi^2(\mathrm{R}) = \sum_{k=1}^{K} \|\mathrm{r}'_k - \mathrm{R}\mathrm{r}_k\|^2 \tag{A.17}$$

Horn proposes an easy to calculate yet robust linear approach to solve this least squares problem by representing the rotation R by a unit quaternion $\mathbf{q}$ and considering the dual problem of maximizing $\sum_{k=1}^{K} {\mathrm{r}'_k}^\mathsf{T}(\mathbf{q}\cdot\mathrm{r}_k\cdot\mathbf{q}^*)$ instead. Throughout this section, quaternions will be represented as 4-vectors in terms of linear algebra as described in section A.3.

The algorithm performs as follows. First, the 3×3-matrix M is computed by:

$$\mathrm{M} = \sum_{k=1}^{K} \mathrm{r}_k {\mathrm{r}'_k}^\mathsf{T} = \begin{pmatrix} M_{xx} & M_{xy} & M_{xz} \\ M_{yx} & M_{yy} & M_{yz} \\ M_{zx} & M_{zy} & M_{zz} \end{pmatrix} \tag{A.18}$$

such that $M_{\mu\nu} = \sum_{k=1}^{K} \mu_k \nu'_k$ for each $\mu, \nu \in \{x, y, z\}$.

Then, the symmetric 4×4-matrix N is computed by linear combinations of the com-

ponents of M:

$$\mathrm{N} = \begin{pmatrix} M_{xx}+M_{yy}+M_{zz} & M_{yz}-M_{zy} & M_{zx}-M_{xz} & M_{xy}-M_{yx} \\ M_{yz}-M_{zy} & M_{xx}-M_{yy}-M_{zz} & M_{xy}+M_{yx} & M_{zx}+M_{xz} \\ M_{zx}-M_{xz} & M_{xy}+M_{yx} & -M_{xx}+M_{yy}-M_{zz} & M_{yz}+M_{zy} \\ M_{xy}-M_{yx} & M_{zx}+M_{xz} & M_{yz}+M_{zy} & -M_{xx}-M_{yy}+M_{zz} \end{pmatrix} \tag{A.19}$$

Now the best fitting rotation R_{opt} corresponds to the unit quaternion vector $\mathbf{q}_{opt}$ which maximizes $\mathbf{q}^{\mathsf{T}}\mathrm{N}\mathbf{q}$. This will be proved further below. Now the unit eigenvector corresponding to the largest eigenvalue λ_{max} of N maximizes $\mathbf{q}^{\mathsf{T}}\mathrm{N}\mathbf{q}$ (note that N is symmetric):

Given the maximum of $\mathbf{q}^{\mathsf{T}}\mathrm{N}\mathbf{q}$ is λ_{max}, there is a unit vector $\mathbf{q}_{max}$ for which holds $\mathbf{q}_{max}^{\mathsf{T}}\mathrm{N}\mathbf{q}_{max} = \lambda_{max} \geq \mathbf{q}^{\mathsf{T}}\mathrm{N}\mathbf{q}$ for all unit vectors $\mathbf{q}$. Hence we have $\mathrm{N}\mathbf{q}_{max} = \lambda_{max}\mathbf{q}_{max}$, i.e. $\mathbf{q}_{max}$ is the unit eigenvector of N corresponding to the eigenvalue λ_{max}, and there are no λ, $\mathbf{q}$ with $\mathrm{N}\mathbf{q} = \lambda\mathbf{q}$ and $\lambda > \lambda_{max}$. So we can identify $\mathbf{q}_{opt}$ with the eigenvector corresponding to the most positive eigenvalue λ_{max} of N. Note that the unit length constraint for $\mathbf{q}_{opt}$ is maintained naturally.

■

Hence the problem of finding the best rotation is reduced to solving an eigenvalue problem $\det(\mathrm{N} - \lambda\mathrm{I}) = 0$. In our implementation, instead of the intricate closed-form solution proposed by *Horn* in [Hor87, 4.C] using *Ferrari's* quartic identity, we use a singular value decomposition to determine the eigenvalues and eigenvectors of N (see app. A.1).

Proof: First, $\phi^2(\mathrm{R}) = \sum_{k=1}^{K} \|\mathrm{r}'_k - \mathrm{R}\mathrm{r}_k\|^2$ becomes minimal iff $\sum_{k=1}^{K} \mathrm{r}'^{\mathsf{T}}_k \mathrm{R}\mathrm{r}_k$ becomes maximal.
This can be easily seen, because for each $k = 1, \ldots, K$ holds:

$$\begin{aligned} \|\mathrm{r}'_k - \mathrm{R}\mathrm{r}_k\|^2 &= (\mathrm{r}'_k - \mathrm{R}\mathrm{r}_k)^{\mathsf{T}}(\mathrm{r}'_k - \mathrm{R}\mathrm{r}_k) \\ &= \mathrm{r}'^{\mathsf{T}}_k \mathrm{r}'_k + (\mathrm{R}\mathrm{r}_k)^{\mathsf{T}}(\mathrm{R}\mathrm{r}_k) - 2(\mathrm{r}'^{\mathsf{T}}_k \mathrm{R}\mathrm{r}_k) \\ &= \|\mathrm{r}'_k\|^2 + \|\mathrm{R}\mathrm{r}_k\|^2 - 2(\mathrm{r}'^{\mathsf{T}}_k \mathrm{R}\mathrm{r}_k) \\ &= 1 + 1 - 2(\mathrm{r}'^{\mathsf{T}}_k \mathrm{R}\mathrm{r}_k) \\ &= 2(1 - \mathrm{r}'^{\mathsf{T}}_k \mathrm{R}\mathrm{r}_k) \end{aligned} \tag{A.20}$$

Therefore, representing the demanded rotation by a unit quaternions $\mathbf{q}$ instead of a rotation matrix R we have to maximize:

$$\begin{aligned}
\sum_{k=1}^{K} \mathrm{r}_k'^{\mathsf{T}}(\mathbf{q}\cdot\mathrm{r}_k\cdot\mathbf{q}^*) &= \sum_{k=1}^{K}(\mathbf{q}^*\cdot\mathrm{r}_k')^{\mathsf{T}}(\mathbf{q}^*\cdot\mathbf{q}\cdot\mathrm{r}_k\cdot\mathbf{q}^*)\\
&= \sum_{k=1}^{K}(\mathbf{q}^*\cdot\mathrm{r}_k')^{\mathsf{T}}(\mathrm{r}_k\cdot\mathbf{q}^*)\\
&= \sum_{k=1}^{K}(\mathbf{T}^*_{\mathrm{r}_k'}\mathbf{q}^*)^{\mathsf{T}}(\mathbf{T}_{\mathrm{r}_k}\mathbf{q}^*)\\
&= \sum_{k=1}^{K}(\mathbf{T}^{*\mathsf{T}}_{\mathrm{r}_k'}\mathbf{q})^{\mathsf{T}}(\mathbf{T}^{\mathsf{T}}_{\mathrm{r}_k}\mathbf{q})\\
&= \sum_{k=1}^{K}\mathbf{q}^{\mathsf{T}}\mathbf{T}^*_{\mathrm{r}_k'}\mathbf{T}^{\mathsf{T}}_{\mathrm{r}_k}\mathbf{q}\\
&= \mathbf{q}^{\mathsf{T}}(\sum_{k=1}^{K}\mathbf{T}^*_{\mathrm{r}_k'}\mathbf{T}^{\mathsf{T}}_{\mathrm{r}_k})\mathbf{q}
\end{aligned} \tag{A.21}$$

where according to the quaternion algebra (see A.3) the left and right multiplication of vectors r_k, r_k' with the unit quaternion $\mathbf{q}$ is defined by the following 4×4-matrices using the framework of linear algebra:

$$\mathbf{T}_{\mathrm{r}_k} = \begin{pmatrix} 0 & -x_k & -y_k & -z_k \\ x_k & 0 & -z_k & y_k \\ y_k & z_k & 0 & -x_k \\ z_k & -y_k & x_k & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{T}^*_{\mathrm{r}_k'} = \begin{pmatrix} 0 & -x_k' & -y_k' & -z_k' \\ x_k' & 0 & z_k' & -y_k' \\ y_k' & -z_k' & 0 & x_k' \\ z_k' & y_k' & -x_k' & 0 \end{pmatrix}$$

By denoting the componentwise products as $M^k_{\mu\nu} := \mu_k\nu_k'$, for each $k = 1,\ldots,K$ and $\mu,\nu \in \{x,y,z\}$ as above, we receive:

$$\mathbf{T}^*_{\mathrm{r}_k'}\mathbf{T}^{\mathsf{T}}_{\mathrm{r}_k} = \begin{pmatrix} M^k_{xx}+M^k_{yy}+M^k_{zz} & M^k_{yz}-M^k_{zy} & M^k_{zx}-M^k_{xz} & M^k_{xy}-M^k_{yx} \\ M^k_{yz}-M^k_{zy} & M^k_{xx}-M^k_{yy}-M^k_{zz} & M^k_{xy}+M^k_{yx} & M^k_{zx}+M^k_{xz} \\ M^k_{zx}-M^k_{xz} & M^k_{xy}+M^k_{yx} & -M^k_{xx}+M^k_{yy}-M^k_{zz} & M^k_{yz}+M^k_{zy} \\ M^k_{xy}-M^k_{yx} & M^k_{zx}+M^k_{xz} & M^k_{yz}+M^k_{zy} & -M^k_{xx}-M^k_{yy}+M^k_{zz} \end{pmatrix}$$

Now define $\mathrm{N}_k = \mathbf{T}^*_{\mathrm{r}_k'}\mathbf{T}^{\mathsf{T}}_{\mathrm{r}_k}$ for each $k = 1,\ldots,K$ and $\mathrm{N} = \sum_{k=1}^{K}\mathrm{N}_k$ and eq. (A.21) turns into $\mathbf{q}^{\mathsf{T}}\mathrm{N}\mathbf{q}$. Using this notation, a rotation R minimizes the squared error function ϕ^2 iff its corresponding unit quaternion $\mathbf{q}$ maximizes $\mathbf{q}^{\mathsf{T}}\mathrm{N}\mathbf{q}$.

By explicitly calculating the elements of N from $M_{\mu\nu}^k$ as implied above one can see that N is in fact identical with the matrix calculated above from the 3×3-matrix M.

Summarizing the proof given, we have shown that the unit quaternion $\mathbf{q}$ which maximizes the term $\mathbf{q}^\mathsf{T}\mathrm{N}\mathbf{q}$ minimizes the squared error function ϕ^2 and hence describes the rotation that fits the data set r_k, r'_k, $k = 1, \ldots, K$ the best.

■

A.5. Method of Lagrangian Multipliers

In mathematical optimization theory, the *method of Lagrangian multipliers*[2] is a common technique to give analytical solutions for optimization problem subject to additional constraints. We will give a short description of the use of Lagrangian multipliers as far as needed for the presented work. For a detailed description of this method we refer to [Arf85].

Consider that we want to maximize the function $f(\mathrm{x})$ subject to K constraints of the form $g_k(\mathrm{x}) = 0$, $k = 1, \ldots, K$. The Lagrangian multiplier method arises from the following geometric interpretation of constrained stationary points, which include constrained extrema of f. Such points must be found on the intersection of the contours of g_k given by $g_k(\mathrm{x}) = 0$ with the image of f where the contour lines for $g_k(\mathrm{x}) = 0$ touch contour lines of f *tangentially*. Computionally, f is normal to the constraints $g_1, \ldots, g_K$ in these points, i.e. x^* is a constrained stationary point iff there are $\alpha_1, \ldots, \alpha_k \neq 0$ such that $\nabla_\mathrm{x} f(\mathrm{x}^*) = \sum_{k=1}^{K} \alpha_k \nabla_\mathrm{x} g(\mathrm{x}^*)$.

Now consider functions $f : \mathbb{R}^n \to \mathbb{R}$, $\mathrm{g} : \mathbb{R}^n \to \mathbb{R}^K$ to be given, and we seek for the maximum of f subject to the constraints $g_k(\mathrm{x}) = 0$ for each $k = 1, \ldots, K$. Define the Lagrangian function Λ with additional unknowns $\lambda \in \mathbb{R}^K$, the *Lagrangian multipliers*, as:

$$\Lambda(\mathrm{x}, \lambda) = f(\mathrm{x}) + \sum_{k=1}^{k} \lambda_k g_k(\mathrm{x})$$

Now f has a constrained optimum in x^* subject to $\mathrm{g}(\mathrm{x}^*) = 0$ iff there is a $\lambda^* \in \mathbb{R}^K$ such that $(\mathrm{x}^*, \lambda^*)$ is a stationary point of Λ.

[2] The term refers to the 18th century mathematician *Joseph Louis Lagrange* who stated the general principle for maximizing a function of n variables subject to additional equations between the variables in his work *"Théorie des Fonctions Analytiques"* (1797).

Proof: For a stationary point $(\mathrm{x}^*, \lambda^*)$ of Λ the gradient $\nabla\Lambda$ evaluates to 0, i.e.:

$$\nabla_{\mathrm{x}}\Lambda(\mathrm{x}^*, \lambda^*) = 0 \Leftrightarrow \nabla_{\mathrm{x}} f(\mathrm{x}^*) = -\sum_{k=1}^{k} \lambda_k^* \nabla_{\mathrm{x}} g_k(\mathrm{x}^*)$$

and

$$\nabla_{\lambda}\Lambda(\mathrm{x}^*, \lambda^*) = 0 \Leftrightarrow \mathrm{g}(\mathrm{x}^*) = 0$$

The first equation denotes f to be to be normal to the constraints in x^*, i.e. x^* to be a constrained stationary point of f. The second equation yields that x^* is valid with respect to the constraints.

■

A.6. Random Sample Consensus

Random sample consensus - abbreviated as RANSAC - as developed by *Fischler and Bolles* [FB81] has become the most common hypothesize-and-test method for robust estimation in presence of outliers arising from incorrect observations. In the RANSAC framework, hypotheses - or *sample solutions* - are generated from a minimal set of observations which is randomly chosen and scored by their fit to the entire set of observations. After the maximal number of iterations has been performed (or for a greedy RANSAC: when a predetermined minimal number of inliers has been reached), the best hypothesis up to date, i.e. the hypothesis that fits the most observations due to some predefined criterion, is taken while the misfit observations are labeled as outliers and are rejected. The RANSAC is considered as "failed" if the best solution does not provide a given minimal number of inliers.

Formally, given a model relating system parameters x to observations y by the implicit constraint $h(\mathrm{x}; \mathrm{y}) = 0$, a RANSAC can be described by a sample solution generator $\hat{\mathrm{x}} = \mathrm{f}(\mathrm{y}_1, \ldots, \mathrm{y}_N)$ which creates a possible solution $\hat{\mathrm{x}}$ from the observation set $\{\mathrm{y}_1, \ldots, \mathrm{y}_N\}$, and an outlier test $b(\mathrm{x}, \mathrm{y}) \in \{0, 1\}$ which decides if an observation y is feasible assuming the system parameters x. When we refer to a RANSAC in practical applications we will declare it by defining a sample solution generator f and an outlier test b. The RANSAC used in our implementation is further parametrized by the maximal number of iterations and the minimal number of inliers needed for success.

B. Appendix II: Implementation

All implementations used for practical applications have been done using the *Basic Image AlgorithmS C++ Library* (BIAS). BIAS has been released by the Multimedia Information Processing group at the Christian-Albrechts-University of Kiel [BIAS] for research and software development of computer vision, motion estimation, and 3d reconstruction algorithms. Additionally, for intrinsic camera calibration existing state-of-the-art software was used. This appendix gives a short overview over aspects of software used for intrinsic camera calibration and rig calibration. Our own work includes mainly the generic rig calibration tool documented in the last section.

Intrinsic Camera Calibration

For intrinsic camera calibration existing software solutions written in Matlab® were used which are regarded as state-of-the-art in common work on the topic.
For intrinsic calibration of perspective cameras and stereo rig calibration in test case 5.5 the Camera Calibration Toolbox developed by *Bouguet* [Bou07] has been used.
For intrinsic calibration of fisheye cameras in test case 5.7 an adaption of the Omnidirectional Camera Calibration Toolbox, originally developed by *Scaramuzza et al.* [SMS06], to the BIAS framework was used.

Pose Estimation Framework

Automatic pose estimation from images is performed by a modular software written in C++ based on the BIAS library referred to as “Tracking Framework”. The theoretical background for pose estimation as implemented in the Tracking Framework is covered in section 3.5. The Tracking Framework supports different feature point detection and matching methods. For this work an implementation of the KLT feature tracking method as described in [TK91] has been used.

Generic Rig Calibration Tool

To perform rig calibration as theoretically outlined in **Chapter 4**, a tool was written in C++ based on the BIAS library. The tool is denoted as "Generic Rig Calibration Tool". The tool operates on lists of time-corresponding projection files or image files containing poses in their meta data as delivered from the Tracking Framework. For output, a rig projection file (described in the XML format used in the BIAS framework) containing the estimated internal poses is created which can be used directly in the Tracking Framework for evaluation. The estimation method can be selected as linear least squares only or linear least squares with additional non-linear refinement. Motion model and scale model are set up manually. Pose correspondences that are determined as misfits to the selected models are removed automatically before estimation as described in sections 4.2.1 and 4.2.2 with heuristics outlined in sections 5.2 and 5.3. The tool is actually available as a command line tool.

Regarding implementational design, we developed a generic class concept that regards our rig parameter estimation methods, i.e. linear least squares approach and non-linear optimization, as realizations of an abstract class of general rig calibration from time-corresponding poses. Although covariances are not considered in our approaches, the interface is designed such that future methods considering covariances can be integrated. Different scale and motion models are considered in the respective classes. The interface of the generic rig calibration class is shown in tab. B.1.

Depending on the motion model, either `Compute` (general motion model) or `ComputeWithoutRotation` (purely translational motion model) are called to estimate the internal poses of the rig. Poses are represented by instances of class `BIAS::PoseParametrization` containing inter alia rotation (represented by a unit quaternion), position and covariances of rotation and position. Time-corresponding input poses $\mathrm{R}_i^k, \mathrm{C}_i^k$ for each time step $k = 1, \ldots, K$ and camera $i = 0, \ldots, N$ are given via parameter *`poses`* containing K vectors of N poses each. Initial solutions for $\Delta\mathrm{R}_i, \Delta\mathrm{C}_i$ which are considered e.g. by the non-linear estimation method are given via parameter *`guess`* containing $N - 1$ poses. Calibration results $\Delta\mathrm{R}_i, \Delta\mathrm{C}_i$ are returned in parameter *`result`*. Internal scales $\Delta\lambda_i^k$ estimated for each time step k are returned in parameter *`scales`*. The interpretation of $\Delta\lambda_i^k$ depends on the scale model used as described in section 4.2.2. The number of internal scales per camera for the time-dependent scale model is set by `SetScalesPerCamera(n)` assuming that the partition of time steps $1, \ldots, K$ into n subsets $\mathcal{K}^{(1)}, \ldots, \mathcal{K}^{(n)}$ is equable, i.e. $|\mathcal{K}^{(\nu)}| = \lceil \frac{K}{n} \rceil$ for each $\nu < n$. `SetScalesPerCamera(1)` defines the time-fixed scale model.

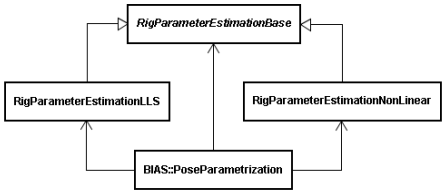


Figure B.1.: Generic rig calibration classes. Class `BIAS::PoseParametrization` from BIAS is used to represent internal and external poses.

RigParameterEstimationBase

`int Compute(`
`in map<BIAS::PoseParametrization> poses,`
`out vector<BIAS::PoseParametrization> result,`
`in vector<BIAS::PoseParametrization> guess,`
`out map<double> scales)`
 	
`int ComputeWithoutRotation(`
`in map<BIAS::PoseParametrization> poses,`
`out vector<BIAS::PoseParametrization> result,`
`in vector<BIAS::PoseParametrization> guess,`
`out map<double> scales)`
 	
`void SetScalesPerCamera(in int numScales)`

Table B.1.: Class interface for generic rig calibration classes. `Compute` performs rig calibration from corresponding poses for the general motion model, `ComputeWithoutRotation` for the purely translational motion model. `SetScalesPerCamera` sets the number of internal scales for each camera, switching between the time-fixed and time-dependent scale model.

C. Acknowledgments and Declaration

Ackknowledgments

I would like to thank Prof. Dr.-Ing. Reinhard Koch for giving me the opportunity to work on such an interesting subject, as well as for his dedicated and instructive mentoring.

Furthermore, I would like to thank the staff of the multimedia information processing group at the Christian-Albrechts-University of Kiel, especially my tutor Felix Woelk for his excellent supervision throughout the entire work on my diploma thesis. This thesis would not have been possible without his great support, theoretical expertise, and constructive comments.

Last but not least, special thanks go to my family, Antje, and my friends, who endured this process with me and offered their help whenever it was possible.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

Sandro Esquivel Olmos, Kiel, 17th September 2007

List of Figures

Bibliography

[Arf85] *G. Arfken*: *"Lagrange Multipliers"*, in: *"Mathematical Methods for Physicists - 3rd Edition"*, p.945–950, Academic Press, Orlando, FL, USA, 1985.

[Boe07] *M. Boettcher*: *"Structure from Motion mit einem starr-gekoppelten Kamerasystem"*, Diploma Thesis, Christian-Albrechts-University, Kiel, 2007.

[Bou07] *J. Y. Bouguet*: *"Camera Calibration Toolbox for Matlab®"*, http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, as at 13th April 2007.

[Che91] *H. Chen*: *"A Screw Motion Approach to Uniqueness Analysis of Head-Eye Geometry"*, in: *"Proceedings Computer Vision and Pattern Recognition"*, p.145–151, Hawaii, USA, June 1991.

[CI01] *Y. Caspi, M. Irani*: *"Alignment of Non-Overlapping Sequences"*, in: *"International Conference on Computer Vision"*, p.76–83, Vancouver, 2001.

[CK91] *J. C. K. Chou, M. Kamel*: *"Finding the Position and Orientation of a Sensor on a Robot Manipulator Using Quaternions"*, in: *"International Journal of Robotics Research, Vol. 10 (30)"*, p.240–254, June 1991.

[FB81] *M. A. Fischler, R. C. Bolles*: *"Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"*, in: *"Communications of the ACM, Vol. 24 (6)"*, p.381–395, 1981.

[HD95] *R. Horaud, F. Dornaika*: *"Hand-Eye Calibration"*, in: *"International Journal of Robotics Research, Vol. 14 (3)"*, p.195–210, 1995.

[MB99] *R. J. Micheals, T. E. Boult*: *"A New Closed Form Approach to the Absolute Orientation Problem"*, Master's Thesis, Lehigh University, 1999.

[MD95] *D. de Menthon, L. S. David*: *"Model-Based Object Pose in 25 Lines of Code"*, in: *"International Journal of Computer Vision, Vol. 15"*, p.123–141, 1995.

[FH86] *O. D. Faugeras, M. Hébert*: *"The Representation, Recognition, and Locating of 3D Objects"*, in: *"International Journal of Robotics Research, Vol. 5 (3)"*, p.27–52, 1986.

[For05] *W. Förstner*: *"Uncertainty and Projective Geometry"*, in: E. Bayro-Corrochano (ed.): *"Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics"*, p.493–535, Springer–Verlag, Berlin, 2005.

[FKK04] *J.-M. Frahm, K. Köser, R. Koch*: *"Pose Estimation for Multi-Camera Systems"*, in: *"Proceedings of the 26th Annual Symposium of the German Association for Pattern Recognition (DAGM '04)"*, p.27–35, Springer–Verlag, Tübingen, 2004.

[FW04] *W. Förstner, B. Wrobel*: *"Mathematical Concepts in Photogrammetry"*, in: *J. C. McGlone, E. M. Mikhail, Roy Mullen* (ed.): *"Manual of Photogrammetry - 5th Edition"*, p.47–49, American Society for Photogrammetry and Remote Sensing, 2004.

[Hor87] *B. K. P. Horn*: *"Closed-Form Solution of Absolute Orientation Using Unit Quaternions"*, in: *"Journal of the Optical Society of America A, Vol. 4 (4)"*, p.629–642, 1987.

[HS93] *R. M. Haralick, L. G. Shapiro*: "Computer and Robot Vision", Addison-Wesley, 1993.

[HS88] *C. Harris, M. Stephens*: *"Combined Corner and Edge Detector"*, in: *"Proceedings of the 4th Alvey Vision Conference"*, p.147–151, 1988.

[HZ00] *R. Hartley, A. Zisserman*: *"Multiple View Geometry in Computer Vision - 2nd Edition"*, Cambridge University Press, Cambridge, 2003.

[Jah97] *B. Jähne*: *"Digitale Bildverarbeitung - 6th Edition"*, Springer–Verlag, Berlin, 2005.

[Koc07] *R. Koch*: *"Method for the Rotation Compensation of Spherical Images"*, Patent Application Publication, No. US2007/0036460 A1, February 2007.

[MNT04] *K. Madsen, H. B. Nielsen, O. Tingleff*: *"Methods for Non-Linear Least Squares Problems - 2nd Edition"*, April 2004.

[Nis03] *D. Nistér*: *"An Efficient Solution to the Five-Point Relative Pose Problem"*, in: *"IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03), Vol. 2"*, p.195, 2003.

[PFTW88] *W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling*: *"Numerical Recipes in C: The Art of Scientific Computing"*, Cambridge University Press, 1988.

[SA89] *Y. C. Shiu, S. Ahmad*: *"Calibration of Wrist-Mounted Robotic Sensors by Solving Homogeneous Transform Equations of the Form $AX = XB$"*, in: *"IEEE Transactions on Robotics and Automation, Vol. 5 (1)"*, p.16–29, February 1989.

[SMS06] *D. Scaramuzza, A. Martinelli, R. Siegwart*: *"A Toolbox for Easy Calibrating Omnidirectional Cameras"*, in: *"Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS '06)"*, Beijing China, October 2006.
http://asl.epfl.ch/∼scaramuz/research/Davide_Scaramuzza_files/Research/OcamCalib_Tutorial.htm, as at 25th May 2006.

[TK91] *C. Tomasi, T. Kanade*: *"Detection and Tracking of Point Features"*, in: *"Carnegie Mellon University Technical Report CMU-CS-91-132"*, April 1991.

[TK92] *C. Tomasi, T. Kanade*: *"Shape and Motion from Image Streams under Orthography: A Factorization Method"*, in: *"International Journal of Computer Vision, Vol. 9 (2)"*, p.137–154, 1992.

[TL89] *R. Y. Tsai, R. K. Lenz*: *"A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration"*, in: *"IEEE Journal of Robotics and Automatation, Vol. 5 (3)"*, p.345–358, June 1989.

[Tsa87] *R. Y. Tsai*: *"A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses"*, in: *"IEEE Journal of Robotics and Automation, Vol. 3 (4)"*, p.323–344, 1987.

[Wan92] *C.-C. Wang*: *"Extrinsic Calibration of a Robot Sensor Mounted on a Robot"*, in: *"IEEE Transactions on Robotics and Automation, Vol. 8 (2)"*, p.161–175, April 1992.

[WEK07] *F. Woelk, S. Esquivel Olmos, R. Koch*: *"Calibration of a Multi-Camera Rig from Non-Overlapping Views"*, in: *"Proceedings of the 29th Annual Symposium of the German Association for Pattern Recognition (DAGM '07)"*, Heidelberg, 2007, to be published.

[WHA93] *J. Weng, T. S. Huang, N. Ahuja*: *"Motion and Structure from Image Sequences"*, Springer–Verlag, Berlin, 1993.

[WR71] *J. H. Wilkinson, C. Reinsch*: *"Linear Algebra. Handbook of Automatic Computation, Vol. II"*, Springer–Verlag, Berlin, 1971.

[Zha99] *Z. Zhang*: *"Flexible Camera Calibration by Viewing a Plane from Unknown Orientations"*, in: *"ICCV"*, p.666–673, 1999.

[ZBR95] *A. Zisserman, P. A. Beardsley, I. D. Reid*: *"Metric Calibration of a Stereo Rig"*, in: *"Proceeding of the IEEE Workshop on Representations of Visual Scenes"*, p.93–100, IEEE Computer Society Press, 1995.

[ZLF96] *Z. Zhang, Q.-T. Luong, O. Faugeras*: *"Motion of an Uncalibrated Stereo Rig: Self-Calibration and Metric Reconstruction"*, in: *"IEEE Transactions on Robotics and Automation, Vol. 12 (1)"*, p.103–113, 1996.

[BIAS] *R. Koch et al.*: *"BIAS - Basic Image AlgorithmS C++ Library"*, available at: http://www.mip.informatik.uni-kiel.de, as at 30th July 2007.