Real-Time Multi-Fisheye Camera Self-Localization and Egomotion Estimation in Complex Indoor Environments

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS (DR.-ING.)

der Fakultät für
Bauingenieur-, Geo- und Umweltwissenschaften
des Karlsruher Instituts für Technologie (KIT)
genehmigte

DISSERTATION

von Dipl.-Ing.

Steffen Urban

aus Worms

Tag der mündlichen Prüfung:
01.12.2016

Referent:	Prof. Dr.-Ing. Stefan Hinz	Institut für Photogrammetrie und Fernerkundung (IPF)
Korreferent:	Prof. Dr.-Ing. Christoph Stiller	Institut für Mess- und Regelungstechnik (MRT)

Karlsruhe 2016

Eidesstattliche Versicherung gemäß §6 Abs. 1 Ziff. 4 der Promotionsordnung des Karlsruher Instituts für Technologie für die Fakultät fÃijr Bauingenieur-, Geo- und Umweltwissenschaften.

1. Bei der eingereichten Dissertation zu dem Thema Real-Time Multi-Fisheye Camera Self-Localization and Egomotion Estimation in Complex Indoor Environments. handelt es sich um meine eigenständig erbrachte Leistung.
2. Ich habe nur die angegebenen Quellen und Hilfsmittel benutzt und mich keiner unzulässigen Hilfe Dritter bedient. Insbesondere habe ich wörtlich oder sinngemäß aus anderen Werken übernommene Inhalte als solche kenntlich gemacht.
3. Die Arbeit oder Teile davon habe ich bislang nicht an einer Hochschule des In- oder Auslands als Bestandteil einer Prüfungs- oder Qualifikationsleistung vorgelegt.
4. Die Richtigkeit der vorstehenden Erklärungen bestätige ich.
5. Die Bedeutung der eidesstattlichen Versicherung und die strafrechtlichen Folgen einer unrichtigen oder unvollständigen eidesstattlichen Versicherung sind mir bekannt. Ich versichere an Eides statt, dass ich nach bestem Wissen die reine Wahrheit erklärt und nichts verschwiegen habe.

Karlsruhe, den 7.10.2016 ______________________________

Abstract

The robust estimation of the camera pose w.r.t observed objects lies at the core of many image-based applications in robotics, photogrammetry and computer vision. In recent years the focus is put more and more on the real-time capability, generality and the possibility to use the developed methods in complex, large-scale indoor and outdoor environments. Especially in challenging indoor scenarios, the field of view of the cameras plays an important role and it is often increased by using fisheye lenses or camera mirror combinations. Apart from increasing the field of view using fisheye lenses, multiple cameras can be rigidly coupled and combined to form a multi-camera system. In the case of multi-camera systems the ambiguities and redundant observations can be effectively used to ensure a robust pose estimation and observed object features are longer visible.

In many applications, the observed object is reconstructed in 3D simultaneously to the estimation of the camera movement. These 3D models, however, are more and more available a-priori, e.g. in form of CAD models. In this case, the reconstruction of the observed object becomes obsolete and an integration of the available 3D information into the egomotion estimation process is required. In addition, the determination of an initial camera pose w.r.t the (local) coordinate frame of the known 3D object is important.

In this thesis, methods for the robust initial and continuous estimation of the pose of a multi-fisheye camera systems are developed. In the first two chapters the emphasis is put on the modeling and description of the multi-camera system as well as the projection relations of fisheye cameras. Then, a method for the compact description of distinct image points, that are used to connect and describe images is presented. It is especially suited for image points detected in wide-angle and fisheye cameras and images with significant distortions. Subsequently, a novel method for the initial pose estimation of a multi-camera system w.r.t 3D building models is presented. In the last chapter, the findings and developed algorithms are combined to a method that is able to robustly estimate the egomotion of a multi-fisheye camera system in real-time in complex environments.

Kurzfassung

Die robuste Bestimmung der Kamerapose relativ zu beobachteten Objekten bildet die Grundlage für viele bildbasierte Anwendungen in Robotik, Photogrammetrie und Computer Vision. Immer häufiger liegt hierbei das Augenmerk auf der Echtzeitfähigkeit, der generellen Anwendbarkeit und der Möglichkeit, die Methoden in großräumigen, komplexen Umgebungen einzusetzen. Dabei spielt der Sichtbereich der Kameras besonders in Innenräumen eine große Rolle. Um diesen zu erhöhen und damit die Robustheit der Algorithmen zu steigern, werden häufig Fisheyekameras oder Kamera-Spiegel Kombinationen verwendet. Eine weitere Möglichkeit ist der Einsatz mehrerer Kameras, die fest miteinander zu einem Kamerasytem verbunden sind. Letzeres hat den Vorteil, dass Mehrdeutigkeiten und redundante Beobachtungen für die robuste Bestimmung ausgenutzt werden können und Merkmale länger verfügbar bleiben.

In vielen Anwendungen wird das beobachtete Objekt während der Bewegung der Kamera, simultan zur Schätzung der Kamerapose, dreidimensional rekonstruiert. Immer häufiger stehen solche 3D Modelle jedoch a-priori in Form von CAD Modellen o.ä. zur Verfügung. Die Rekonstruktion des Objekts ist in solchen Fällen obsolet und eine geschickte Einbindung des Modells in die Methoden zur Bestimmung der Eigenbewegung der Kamera werden nötig. Des Weiteren ist die Schätzung einer initialen Kamerapose relativ zum bekannten (lokalen) Koordinatensystem des Objekts von Bedeutung.

In dieser Arbeit werden Methoden für die robuste initiale und kontinuierliche Schätzung der Pose eines Multi-Fisheye Kamerasystems entwickelt. Der Schwerpunkt der ersten beiden Kapitel liegt zunächst bei der Modellierung des Multi-Kamerasytems und den Abbildungseigenschaften der Fisheyekameras. Anschließend wird ein neues Verfahren für die kompakte Beschreibung von distinkten Bildpunkten, die für die Verknüpfung und Beschreibung von Bildern verwendet werden vorgestellt, dass besonders für Bilder mit starker Verzeichnung bzw. anderen Projektionsvorschriften geeignet ist. Das darauffolgende Kapitel stellt einen neuen Ansatz zur Schätzung der initialen Pose des Multi-Fisheye Kamerasystems in 3D Modellen vor. Im letzten Kapitel werden die Erkenntnisse und entwickelten Methoden zu einem neuen Verfahren für die Schätzung der Eigenbewegung des Multi-Fisheye Kamerasystems in Echtzeit in komplexen Umgebungen kombiniert.

Contents

Acronyms **7**

Symbols **9**

1 Introduction **1**
1.1 Problem Statement and Motivation . . . 1
1.2 Open Challenges and Contributions . . . 2
1.3 Outline . . . 4

2 Principles, Mathematical Models and Datasets **5**
2.1 Iterative Least-Squares Optimization . . . 5
2.2 Camera Pose . . . 6
2.2.1 Rotation Parametrization . . . 7
2.2.1.1 Euler Angles . . . 7
2.2.1.2 Quaternions . . . 8
2.2.1.3 Cayley . . . 8
2.2.1.4 Exponential Map . . . 8
2.2.1.5 Speed and Convergence . . . 9
2.3 Fisheye Cameras - Imaging Process . . . 10
2.4 Computer Graphics . . . 11
2.5 Datasets and Multi-Camera System . . . 12
2.6 Summary . . . 14

3 Improved Omnidirectional Camera Calibration **15**
3.1 State-of-the-Art . . . 16
3.2 Camera Model . . . 17
3.3 Original Calibration Methodology . . . 17
3.3.1 Linear Estimation of the Exterior and Interior Orientation . . . 17
3.3.2 Center of Distortion and Linear Refinement . . . 18
3.3.3 Two-Step Non-linear Refinement . . . 18
3.4 Analysis and Contributions . . . 18
3.5 Results and Comparisons . . . 20
3.6 Summary . . . 23

4 MultiCol - Bundle Adjustment for Multi-Camera Systems **25**
4.1 Principles of Multi-Camera Systems and Related Work . . . 25
4.2 The MultiCol Model . . . 27
4.2.1 Classical Collinearity Equations . . . 27
4.2.2 Expanded Collinearity Equations . . . 28
4.2.3 Hyper Graph Modeling . . . 30
4.3 Difference to Other Models . . . 31
4.4 MCS Calibration . . . 33
4.5 Experiments and Results . . . 33
4.5.1 Simulations . . . 34
4.5.2 Interior Self-Calibration . . . 38
4.5.3 Real Data . . . 40
4.6 Summary . . . 43

5 Online Adapted Binary Descriptors for Omnidirectional Images **45**
5.1 State-of-the-Art 46
5.1.1 Keypoint Detectors 46
5.1.2 Keypoint Descriptors 48
5.2 dBRIEF - Distorted BRIEF 50
5.3 Simulations 52
5.4 Improving dBRIEF by Online Mask Learning 55
5.4.1 Offline Learning 58
5.4.2 Online Mask Learning 59
5.5 Results on Real Data 60
5.6 Summary 63

6 Multi-Camera System Self-Localization in Textureless Building Models **65**
6.1 State-of-the-Art 65
6.2 Model-Based Self-Localization 67
6.2.1 MCS Pose Sampling 67
6.2.2 Rendering Synthetic Views 68
6.2.3 Descriptor Extraction 69
6.2.4 Matching 70
6.2.5 Nearest Neighbor Initialization 71
6.2.6 Particle Filter Initialization 71
6.3 Experiments and Results 73
6.3.1 Simulations 73
6.3.2 Real World Ground Truth 75
6.4 Summary 79

7 MultiCol-SLAM - A Multi-Fisheye SLAM System **81**
7.1 SLAM - Principles and State-of-the-Art 81
7.1.1 Keyframe-Based Approaches 83
7.1.2 Model-based Tracking 85
7.1.3 Open Challenges and Contributions 86
7.2 Framework 86
7.2.1 Map Entities 87
7.2.2 Tracking Thread 89
7.2.3 Mapping Thread 91
7.2.4 Integration of 3D Model Information 94
7.2.5 Loop Detection and Closing Thread 96
7.3 Experiments and Results 99
7.3.1 Single- vs. Multi-Camera SLAM 101
7.3.2 Impact of Features 102
7.3.3 Model-Supported Tracking 102
7.4 Summary 104

8 Conclusion and Outlook **105**
8.1 Summary and Conclusion 105
8.2 Outlook 106

Bibliography **109**

A Acknowledgements **123**

Acronyms

ATE	Absolute Trajectory Error.
AGAST	Adaptive and Generic corner detection based on the Accelerated Segment Test.
API	Application Programming Interface.
BoW	Bag-of-Words.
BOLD	Binary Online Learned Descriptor.
BRIEF	Binary Robust Independent Elementary Features.
BRISK	Binary Robust Invariant Scalable Keypoints.
BA	Bundle Adjustment.
BACS	Bundle adjustment for multi-camera systems.
CNN	Convolutional Neural Network.
DoF	Degrees of Freedom.
FLANN	Fast Library for Approximate Nearest Neighbor search.
FAST	Features from Accelerated Segment Test.
FPGA	Field Programmable Gate Array.
FoV	Field of View.
FPS	Frames per Second.
GPS	Global Positioning System.
IMU	Inertial Measurement Unit.
LDA	Linear Discriminant Analysis.
ML	Maximum Likelihood.
MSRE	Mean Squared Reprojection Error.
MAV	Micro Aerial Vehicle.
MCS	Multi-Camera System.
MKF	Multi-Keyframe.
NPnP	Non-Perspective-N-Point problem.
ORB	Oriented FAST and Rotated BRIEF.
PTAM	Parallel Tracking and Mapping.
PnP	Perspective-N-Point problem.
pdf	Probability Density Function.
RANSAC	Random Sample Consensus.
RPE	Relative Pose Error.
RMSRE	Root Mean Squared Reprojection Error.
SIFT	Scale Invariant Feature Transform.
SLAM	Simultaneous Localization and Mapping.
SVP	Single Effective Viewpoint.
SURF	Speeded Up Robust Features.
SfM	Structure From Motion.
SSRE	Sum of Squared Reprojection Errors.

Symbols

a,$\mathbf{a}$,$\mathbf{A}$	Scalar, vector, matrix.
$\hat{a}$,$\hat{\mathbf{a}}$,$\hat{\mathbf{A}}$	Estimated quantities.
$\mathbf{v}$	Camera ray or bearing vector.
ω	Rodriguez vector.
$\mathbf{n}$	Normal vector.
$\mathbf{m}'$	Point in the image plane.
u',v'	Point coordinates in the image plane.
$\mathbf{m}$	Point in the sensor plane.
u,v	Point coordinates in the sensor plane.
$\mathbb{N}$,$\mathbb{Z}$,$\mathbb{R}$,$\mathbb{H}$	Natural, integer, real numbers, Hamilton algebra.
$\mathbf{I}$	Identity matrix.
$\mathbf{J}$	Jacobian matrix.
$\mathbf{R}$	3×3 rotation matrix.
$\mathbf{t}$	3×1 translation vector.
$\mathbf{M}$	homogeneous transformation matrix.
$\mathbf{q}$	Minimal representation vector of transformation matrix.
$\mathbf{S}$	Similarity transformation matrix.
$\mathbf{H}$	Homography.
$\mathbf{F}$	Fundamental matrix.
$\mathbf{E}$	Essential matrix.
$\mathbf{P}$	Information matrix.
$\mathbf{p}$	3D point.
$\mathbf{p}_c$,$\mathbf{p}_w$	3D point in camera and world coordinates respectively.
$\mathbf{o}$	Principal point.
ρ	Radial distance from principal point.
θ	Incidence angle. measured to z-axis of camera frame.
$f(\rho)$	Forward projection polynomial. camera to world.
$\rho(\theta)$	Backward projection polynomial. world to camera.
$\mathbf{\Phi}$	Orientation descriptors.
$\mathbf{Q}$,$\mathbf{Q}_d$,$\mathbf{Q}_f$	Set of binary tests, distorted tests, fisheye projected tests.
ζ	Minimal 6 DoF pose vector, containing the Rodrigues vector ω and the translation vector $\mathbf{t}$.
e	Huber tuning constant.
χ	Co-visibility weight between two Multi-Keyframes.

1. Introduction

Recovering the pose of a camera with respect to the world it observes only from the images it acquires is a challenge that motivated researchers for decades to develop methods that cover all aspects of modern photogrammetry, robotics and computer vision. Starting from the mapping relations that model the projection of a three dimensional world point to the two dimensional image plane, to geometric algorithms and mathematical descriptions that allow to describe the mutual position between the camera and the world it observes as well as between multiple cameras. After having established most of the basic geometric methods, the next challenge was to exploit that knowledge to develop automatic and robust algorithms that could determine those relations from images alone. Hence, corresponding image points needed to be found across different camera poses and consequently algorithms emerged that are able to extract distinct and repeatable image points in multiple images. Most methods transform the radiometry in the vicinity of a measured image point into a robust description that is invariant under camera pose changes.

With increasing computing capacity, especially for mobile computers, the aforementioned fundamentals were connected and transformed into real-time approaches that measure image points, connect and match these to prior measurements and hereby estimate the trajectory of a video camera at frame rate. In this context the terms Structure From Motion (SfM) and Simultaneous Localization and Mapping (SLAM) prevailed. In both definitions the camera reconstructs the geometry of the world it observes whilst simultaneously recovering its pose.

Nowadays the motivation to robustly estimate the camera pose not only for continuous video streams over time but also to localize the camera in environments it has not seen before is stronger than ever. With mobile devices having plenty of computing power readily available, new applications and opportunities for mobile camera systems open up, such as augmenting the reality with virtual content that is projected to the images, making cars drive autonomously, helping visually impaired people or simply pointing someone in the right direction.

In the following sections, the problem as well as the motivation of the thesis is stated. Subsequently, objectives and contributions are derived and the remainder of the thesis is outlined.

1.1 Problem Statement and Motivation

This work is structured in a similar fashion and faces the same challenges as the endeavor recapitulated in the introduction. At its core lies the goal to robustly estimate the trajectory of a multi-fisheye camera system and to develop an approach for the self-localization of its pose from building models it has not necessarily seen before. At a more abstract level, the work is embedded in the effort to construct an augmented reality system for underground construction sites. The system is supposed to allow planners to compare, document and inspect multi-scale 3D building models and construction plans on-site and during the building phase. Especially in such scenarios, with difficult illumination conditions, narrow and obstructed paths, moving and dynamically changing objects, robust, vision based algorithms are needed. In addition, external positioning systems such as Global Positioning System (GPS) or indoor navigation is either not available or cumbersome to install and calibrate with frequently changing conditions. The methods and algorithms developed in this thesis, however, are all general and not limited to the scenario described above.

Hence, in this work, a multi-fisheye camera system based SLAM approach is developed. Using wide-angle or fisheye cameras ensures that visual clues remain visible in the camera over a longer period of time and thus help to robustly estimate the camera pose even in narrow scenes. By coupling multiple cameras, a 360° panorama of the environment is maintained and the pose can be recovered even if one camera image is completely occluded.

An additional challenge is the efficiency of the methods. Most applications require the system to operate in real-time, i.e. at frame rate. Using multiple cameras, however, multiplies the computational burden and although mobile computing power steadily increases, it is quickly stretched to its limits. Thus apart from robustness, the focus has to be put on the efficiency and runtime of methods.

As accurate multi-scale 3D models are more and more available not only in the context of the construction site project, but also for cities and public buildings, an integration of such information into the multi-camera SLAM pipeline is desirable. Latter introduces some additional challenges, such as where and when to integrate the information, the projection into fisheye cameras but is also renders the self-localization in unseen environments and the robust re-localization in changing conditions possible.

1.2 Open Challenges and Contributions

Although researchers have developed outstanding methods over the past years in most aspects of this thesis, i.e. camera calibration, multi-camera pose estimation, re-localization, SfM and SLAM, some open challenges can be identified:

Arbitrary multi-camera systems

Challenge: Apart from stereo systems, where similar cameras with almost equal viewpoint and direction are rigidly coupled, Multi-Camera System (MCS)s are often mentioned in the context of motion capture systems, where multiple cameras observe a common volume. These motion capture systems are usually composed of the same camera type, are static and reconstruct an object that is moving in the observed volume. Rigidly coupled cameras with non-overlapping Field of View (FoV) became more popular in recent years as they promise panoramic views of the environment. Still, coupling arbitrary cameras with potentially varying mapping relations (perspective and fisheye) is challenging and cumbersome. In addition it would be convenient to have a method that can cope with reconstruction, pose estimation and calibration at the same time.

Contribution: In this work, we extend the common collinearity equations for multi-camera systems. First, the MCS to camera transformations are integrated. Using a generic interior orientation parametrization for the world to camera projection of each camera, the formulation can then be used to estimate the pose of the MCS, calibrate the mutual camera transformations and reconstruct the scene.

Multi-fisheye camera SLAM

Challenge: Many monocular, stereo and RGB-D SLAM approaches exist and promising systems were released open source in recent years. They integrate research from many directions of computer vision, photogrammetry and robotics research such as direct pose estimation, robust statistics, feature extraction, projective geometry, bundle adjustment, place recognition and loop closing. Yet complete MCS SLAM approaches are rare, as most methods need to be extended, made more efficient and many direct geometric methods for MCSs were just published in the last two years. In addition, the system has to be efficiently separated into smaller problems, that can be executed in parallel in multiple threads.

Contribution: Based on the generic formulation of the collinearity equations and a monocular SLAM system, we develop a multi-fisheye camera SLAM system, that is complete in the sense that it contains all essential building blocks, i.e. pose estimation, reconstruction, re-localization and loop closing.

Efficient, robust features for non-perspective cameras

Challenge: Extracting and matching robust, distinct features either between image pairs or hundreds of images is a vast area of research. Image point descriptors can be roughly categorized into being binary or real-valued. Although real-valued descriptors have a slightly better matching performance, binary descriptors outperform them in terms of speed and storage cost. So far, all binary descriptors assume the pixels to be the result of a perspective camera mapping which is not the case for cameras that have large distortion or completely different mapping relations such as fisheye cameras.

Contribution: Instead of re-sampling each image into a perspective one, which is costly and introduces artifacts, we propose a distorted version of a popular binary descriptor. Our representation relies on a calibrated camera but is still fast to compute and maintains the advantages of compact binary descriptors that are fast to match.

Online adaption of features

Challenge: Most feature-based SLAM systems store a descriptor for each point they reconstruct. This descriptor can then be matched to point descriptions in the current frame. These fixed descriptors might change over time, e.g. due to changing lightning conditions, or slightly wrong point extractions. In addition most state-of-the art descriptors are learned and tested on specific datasets, that do not necessarily contain similar characteristics as the scene that is currently observed. Furthermore the learning is slow and done off-line.

Contribution: In this thesis, we extend an online learning method for binary features. Here binary masks are learned along with each descriptor by simulating radiometric deformations on-line. We will adapt this method and use it in conjunction with distorted descriptors.

Initialization, self-localization

Challenge: Initialization and self-localization can be seen from different perspectives. If the camera (-system) already visited a scene, place recognition or exhaustive matching schemes together with statistic sampling methods can be employed to derive an initial pose from the existing map. If such a pre-build model is not available, the first crucial step for all SLAM systems is the map initialization. Monocular SLAM systems usually initialize a model by triangulation from two images. Here the difficulty is to find heuristics and to decide whether the scene and/or the camera movement allows for 3D reconstruction. The self-localization of cameras w.r.t. untextured building models is a topic that is rarely addressed.

Contribution: In this thesis, we propose two different localization methods. The first is the re-localization of a MCS either after tracking loss or if the systems re-visits a place. The second is a self-localization method using untextured building models. By comparing features from real and synthetic view of the environment, the system is able to estimate its pose w.r.t the building model either using approximate nearest neighbor matching or particle filtering.

Model-supported multi-camera SLAM

Challenge: In literature, various model-based tracking methods exist. Most work on edges that are extracted from rendered versions of the model and are compared to image edges. Some also extract points on the model and perform a fusion of edge and point information. These methods are applied to workspace sized scenarios and the 3D models are usually quite small and have only few surfaces.

Contribution: In this thesis, we integrate 3D models the size of buildings and even construction sites into the SLAM pipeline. A method is proposed that renders the model from distinct camera poses (Keyframes) and aligns these the 3D model to image edges. Thus it supports the SLAM trajectory and prevents the the continuous MCS pose estimates from drift.

1.3 Outline

The thesis is structured as follows. Chapter 2 gives an introduction into camera pose parametrization, the mapping relations from world to image points and the basics of computer graphics needed in this thesis. In addition, the datasets that are used in this thesis are subsumed. Subsequently, Chapter 3 introduces the camera model and an improved calibration methodology. In Chapter 4, the camera model is then used in conjunction with extended collinearity equations to model arbitrary multi-camera systems and various synthetic and real world experiments are performed to evaluate the performance of the formulation. In Chapter 5 binary features are proposed, that take the distortion introduced by fisheye lenses into account. In addition an online adaptable version is presented and evaluated. Chapter 6 contains a model-based self-localization methodology for multi-fisheye camera systems based on rendered fisheye views of a large 3D model and edge features. Chapter 7 combines all chapters into a multi-camera SLAM system that is evaluated and tested in various scenarios. Finally, Chapter 8 summarizes and concludes the thesis followed by some ideas for potential future work.

2. Principles, Mathematical Models and Datasets

This chapter gives an overview of models, concepts, mathematical tools, computer graphics and all datasets that are used and needed throughout this thesis. First the important aspects of iterative non-linear least-squares optimization methods are explained, as they are one of the most fundamental tools for parameter estimation in the context of camera calibration, pose estimation and reconstruction. Subsequently, the concept and parametrization of the camera pose is outlined. In addition, we focus on rotation parametrization as it is crucial for efficient parameter estimation. Then, the geometrical mapping from the three dimensional world to images is detailed with a focus on the differences between classical perspective and fisheye cameras. Afterwards, the elementary parts of the computer graphics Application Programming Interface (API) used in this work are presented which is used to render synthetic but geometrically correct fisheye images of building models. Finally, all multi-camera datasets used in this thesis are briefly introduces as they are frequently used in most chapters.

2.1 Iterative Least-Squares Optimization

In this section we recapitulate important aspects of (non-linear) least-squares optimization, as it is frequently used throughout this thesis (see Chapter 3,4,7). More in-depth introductions can be found in [73, 116, 140, 177].

Let $\mathbf{m}'$ be a measurement and $\hat{\mathbf{m}}'(\mathbf{x})$ a corresponding measurement function that depends on a vector of parameters $\mathbf{x}$. The difference between both is given by the residual vector $\mathbf{r} = \mathbf{m}' - \hat{\mathbf{m}}'(\mathbf{x})$. In general, we are looking for the parameters $\mathbf{x}$ that best describe our measurements, or in other words we are looking for the maximum likelihood (ML) distribution $p(\mathbf{m}'|\mathbf{x})$ of our measurements, given the parameters. Assuming a Gaussian distribution, this can be expressed up to proportionality by:

$$p(\mathbf{m}'|\mathbf{x}) \propto \exp\left[(\mathbf{m}' - \hat{\mathbf{m}}'(\mathbf{x}))^T \mathbf{P} (\mathbf{m}' - \hat{\mathbf{m}}'(\mathbf{x}))\right] \tag{2.1}$$

Maximizing the likelihood is equivalent to minimizing the negative log-likelihood (see also [143]) and thus:

$$\min_{\mathbf{x}} - \log p(\mathbf{m}'|\mathbf{x}) = \min_{\mathbf{x}} (\mathbf{m}' - \hat{\mathbf{m}}'(\mathbf{x}))^T \mathbf{P} (\mathbf{m}' - \hat{\mathbf{m}'}(\mathbf{x})) = \min_{\mathbf{x}} \mathbf{r}^T \mathbf{P} \mathbf{r} \tag{2.2}$$

where $\mathbf{P}$ is the information matrix and $\mathbf{r}$ is the vector of residuals. Assuming uncorrelated observations, $\mathbf{P}$ is block diagonal and Equation 2.2 simplifies to a sum of squares:

$$\min_{\mathbf{x}'} \sum_i (\mathbf{m}'_i - \hat{\mathbf{m}}'_i(\mathbf{x}))^T \mathbf{P}_i (\mathbf{m}'_i - \hat{\mathbf{m}}'_i(\mathbf{x})) = \min_{\mathbf{x}} \sum_i \mathbf{r}^T \mathbf{P} \mathbf{r} \tag{2.3}$$

So far, a linear relation between measurements and parameters is assumed. In general $\hat{\mathbf{m}}'(\mathbf{x})$ is a non-linear function and has to be linearized around an initial estimate of $\mathbf{x}_0$. The partial derivative of each measurement i w.r.t each parameter j is given by $J_{ij} = \frac{\partial \hat{m}'_i}{\partial x_j}$. The corresponding matrix $\mathbf{J_x} = \frac{\partial \mathbf{m}'}{\partial \mathbf{x}}$ is called the Jacobian. Now, Gauss-Newton updates can be used to minimize the cost (sum of squares Equation 2.3) by iteratively updating the initial parameter vector $\mathbf{x}_{s=0}$:

$$\mathbf{x}_{s+1} = \mathbf{x}_s + d\mathbf{x} \tag{2.4}$$

at update step $s = 1, .., S$. Starting from a good initial estimate, convergence is usually achieved in few iterations ($S < 20$). The update vector $d\mathbf{x}$ is found by solving:

$$(\mathbf{J}_\mathbf{x}^T \mathbf{P} \mathbf{J}_\mathbf{x}) d\mathbf{x} = -\mathbf{J}_\mathbf{x}^T \mathbf{P} \mathbf{v} \tag{2.5}$$

which are also called the normal equations. However, in this thesis, the augmented normal equations are used and Levenberg-Marquardt [112] iterations are performed:

$$(\mathbf{J}_\mathbf{x}^T \mathbf{P} \mathbf{J}_\mathbf{x} + \lambda \mathbf{I}) d\mathbf{x} = -\mathbf{J}^T \mathbf{P} \mathbf{v} \tag{2.6}$$

where $\mathbf{I}$ is a identity matrix and λ is a scaling factor. If any parameter update step (Equation 2.4) leads to a reduction of the error, λ is divided by some factor, i.e. its value decreases. Thus for small λ the augmented normal equations essentially perform the same Gauss-Newton update as estimated from Equation 2.5. However if the update step leads to an error increase, λ is multiplied by some factor the normal equation matrix is approximated by $\lambda \mathbf{I}$ performing gradient descent updates. Another interpretation is that, the parameter λ regulates the strength and the direction of the update. Gauss-Newton updates usually lead to fast convergence in the vicinity of the solution, whereas Gradient Descent updates make smaller steps and will eventually decrease the error for larger λ as smaller steps are performed.

Thus far, we assumed that the data is free of outliers, i.e. wrong measurements. In the context of this work such outliers, however, very frequently impair the data and hence would disturb the optimization significantly. Thus, usually iterative re-weighted least squares adjustment (IRLS) is conducted using a robust cost function (M-estimator). Instead of minimizing the squared residuals (Equation 2.3), a function ζ is used that re-weights outliers:

$$\min_{\mathbf{x}} \sum_i \zeta(\mathbf{m}'_i - \hat{\mathbf{m}}'_i(\mathbf{x})) = \min_{\mathbf{x}} \sum_i \zeta(\mathbf{r}_i) \tag{2.7}$$

In this thesis we use the Huber estimator, but many others exist [140]:

$$\zeta_e(\mathbf{r}) = \begin{cases} 0.5\mathbf{r}^2, & |\mathbf{r}| \le e \\ e/|\mathbf{r}| - 0.5^2, & |\mathbf{r}| > e \end{cases} . \tag{2.8}$$

where $e = 1.345$ is the tuning constant.

2.2 Camera Pose

The camera pose is defined by a rotation $\mathbf{R} \in SO(3)$ and a position $\mathbf{t}_0 \in \mathbb{R}^3$ relating the camera to a fixed world frame. $SO(3)$ represents the Lie group of rotations matrices in $\mathbb{R}^3$ (see [58]). Both can be composed into a 4×4 homogeneous transformation matrix $\mathbf{M}$:

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t}_0 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x_0 \\ r_{21} & r_{22} & r_{23} & y_0 \\ r_{31} & r_{32} & r_{33} & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

representing the camera to world frame transformation. The inverse, world to camera frame transformation of $\mathbf{M}$ is given by:

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t}_0 \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.10}$$

where we made use of the fact that rotation matrix $\mathbf{R}$ is orthogonal and thus $\mathbf{R}^T = \mathbf{R}^{-1}$. Now, given a point $\mathbf{p}_w = [x_w, y_w, z_w, 1]^T$ in homogeneous coordinates in the world frame, its transformation to the camera frame is given by:

$$\mathbf{p}_c = \mathbf{M}^{-1} \mathbf{p}_w = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t}_0 \\ \mathbf{0}^T & 1 \end{bmatrix} [x_w, y_w, z_w, 1]^T \tag{2.11}$$

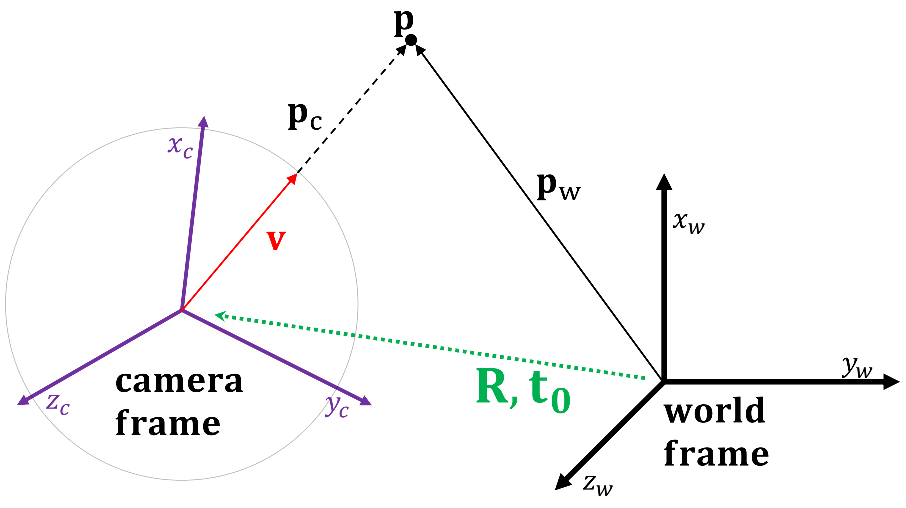


Figure 2.1. The camera pose is defined by a rotation $\mathbf{R}$ and translation $\mathbf{t}_0$. The world point $\mathbf{p}_w$ can be rotated to the camera frame using Equation 2.11. The vector $\mathbf{v}$ is a normalized version of $\mathbf{p}_c$ that points in the same direction. This vector is also called camera ray or bearing vector.

Normalizing this vector to unit length yields a bearing vector (or camera ray):

$$\mathbf{v} = \frac{\mathbf{p}_c}{\|\mathbf{p}_c\|} \tag{2.12}$$

In addition, the relative transformation (orientation) between two camera poses $\mathbf{M}_1$ and $\mathbf{M}_2$ can be easily computed using homogeneous matrices:

$$\mathbf{M}_{21} = \mathbf{M}_2^{-1}\mathbf{M}_1 \tag{2.13}$$

Although the homogeneous representation represents a practicable way of transforming points between rigid coordinate frames and computing relative transformations, it contains more parameters than necessary. In its minimal form a rigid body transformation in $\mathbb{R}^3$ has six degrees if freedom, i.e. three offsets and three rotations about each coordinate axis. The three offsets are already well encoded by the translation component $\mathbf{t}_0$. The rotation matrix $\mathbf{R}$ however contains nine instead of three elements and a minimal parametrization is needed. Otherwise the orthogonality constrains had to be enforced, i.e. if $\mathbf{R}$ is incrementally updated during optimization. As the pose parameters are frequently subject to optimization the subsequent sections deal with minimal parametrization of rotations.

2.2.1 Rotation Parametrization

In this section, several minimal rotation parametrizations are presented that are usually applied and used in different contexts. In addition, we contrast their advantages and disadvantages for the given tasks, e.g. frequent pose estimation, bundle adjustment and loop closing for multi-camera systems.

2.2.1.1 Euler Angles

Let ω, ϕ, κ be the three Euler angles that represent a rotation of the coordinate frame about the respective coordinate axis $\mathbf{x}, \mathbf{y}, \mathbf{z}$, i.e. $\mathbf{R_x}(\omega), \mathbf{R_y}(\phi), \mathbf{R_z}(\kappa)$. Then a rotation matrix that represents a counter-clockwise rotation about $\mathbf{x}$, then $\mathbf{y}$ and finally $\mathbf{z}$ is given by:

$$\mathbf{R_{x,y,z}}(\omega,\phi,\kappa) = \begin{bmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix} \tag{2.14}$$

Note that the sequence in which rotations about the axes are applied matters. The inverse mapping from the given rotation matrix to Euler angles is given by:

$$\begin{bmatrix} \omega \\ \phi \\ \kappa \end{bmatrix} = \begin{bmatrix} atan2(-r_{2,3}, r_{3,3}) \\ -asin(r_{1,3}) \\ atan2(-r_{1,2}, r_{1,1}) \end{bmatrix} = \begin{bmatrix} acos(r_{3,3}/\cos\phi) \\ -asin(r_{1,3}) \\ acos(r_{1,1}/\cos\phi)) \end{bmatrix} \tag{2.15}$$

where $atan2()$ is a function that considers the quadrant. Again, the inverse mapping also depends on the rotation sequence. In addition the angles can not be determined uniquely if the second rotation (about ϕ) is $\phi = \pi/2 + n\pi$ for $n \in \mathbb{Z}$. Apart from depending on rotation sequence, the singularities of Euler angles can not be avoided during optimization. Compared to a quaternion representation, Euler angles are also less accurate when the rotation is incrementally updated over time or interpolated. An advantage of Euler angles is that they are easy to use and implement and give an intuitive understanding of a rotation in $\mathbb{R}^3$.

2.2.1.2 Quaternions

Quaternions are less intuitive, but avoid the deficiencies of Euler angles. Let $\mathbf{q} \in \mathbb{H}$ be a quaternion represented by a vector:

$$\mathbf{q} = [w, q_1, q_2, q_3]^T \tag{2.16}$$

To represent a rotation the constrain $\|\mathbf{q}\| = 1$ has to hold, i.e. $\mathbf{q}$ needs to be a unit quaternion. A rotation about the unit vector ω by angle α is given by the unit quaternion:

$$\mathbf{q} = \cos(\alpha/2) + \omega \sin(\alpha/2) \tag{2.17}$$

and the rotation of a point $\mathbf{p}$ is given by:

$$\mathbf{p}_r = \mathbf{q} * \mathbf{p} * \bar{\mathbf{q}} \tag{2.18}$$

where $\bar{\mathbf{q}}$ is the adjoint and $*$ quaternion multiplication. Mapping a quaternion to a rotation matrix and back is straightforward but the equations are omitted here for readability. The reader is referred to [42]. Although quaternions avoid any singularities and allow for accurate interpolation between rotations, they have other drawbacks. They are over-parametrized, as they have four degrees of freedom. This is a overhead we would like to avoid during optimization where a minimum number of parameters is crucial. In addition the unity norm constrain has to be enforced during optimization somehow (see [161]). Finally, chaining rotations is faster with quaternions, but rotating vectors needs more than twice as much operations and is thus slower (see [19]). Latter however is done a lot more frequently in the context of this work (e.g. rotating points to the camera frame and back). There exists an extension to quaternions which is called dual quaternions [84]. It offers a unified expression for translation and rotation using dual number theory. It combines two quaternions called the real and the dual part. Again this representation is over-parametrized as it contains eight degrees of freedom and has to be re-normalized after each iteration in a optimization process.

2.2.1.3 Cayley

The Cayley transform maps between skew-symmetric and special orthogonal matrices. Using this transformation, we can map a rotation matrix to its minimal vector $\mathbf{c} = [c_1, c_2, c_3]^T$:

$$\mathbf{C}(\mathbf{R}) = \begin{bmatrix} 0 & -c_3 & c_2 \\ c_3 & 0 & -c_1 \\ -c_2 & c_1 & 0 \end{bmatrix} = (\mathbf{R} - \mathbf{I})(\mathbf{R} + \mathbf{I})^{-1} \tag{2.19}$$

Setting one of the elements of $\mathbf{c}$ to one and the others to zero results in a $\pi/2$ rotation about that respective axis. The mapping from the minimal representation to a rotation matrix is as follows:

$$\mathbf{R}(\mathbf{c}) = \frac{1}{s} \begin{bmatrix} 1 + c_1^2 - c_2^2 - c_3^2 & 2(c_1c_2 - c_3) & 2(c_1c_3 + c_2) \\ 2(c_1c_2 + c_3) & 1 - c_1^2 + c_2^2 - c_3^2 & 2(c_2c_3 - c_1) \\ 2(c_1c_3 - c_2) & 2(c_2c_3 + c_1) & 1 - c_1^2 - c_2^2 + c_3^2 \end{bmatrix} \tag{2.20}$$

with $s = 1 + c_1^2 + c_2^2 + c_3^2$. From Equation 2.19 and Equation 2.20 it can be seen, that the conversion in both directions will be very efficient, as both equations are free of trigonometric functions and have only very few operations. The Cayley representation has singularity at π.

2.2.1.4 Exponential Map

In this representation the rotation is expressed by an axis of rotation $\omega = [\omega_x, \omega_y, \omega_z]^T$ and an angle $\alpha = \|\omega\|$, thus it is also known as the axis-angle representation. Given the skew-symmetric matrix version of ω:

$$[\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.21}$$

where $[]_\times$ is an operator that maps a 3-vector to its skew symmetric form. The mapping from ω to its corresponding rotation matrix is given by:

$$\mathbf{R}_{rod}(\omega) = \exp_{SO3}(\omega) = \mathbf{I}_{3\times 3} + \frac{\sin(\alpha)}{\alpha}[\omega]_\times + \frac{1 - cos(\alpha)}{\alpha^2}[\omega]_\times^2 \tag{2.22}$$

which is known as the Rodrigues' formula. The division by zero has to be avoided by testing if the norm of ω vanishes. The axis-angle representation allows to express a rotation with the minimal number of parameters and is free of constrains. Thus it is well suited for unconstrained optimization algorithms.

Expressing the homogeneous representation from Equation 2.9 using $\mathbf{R}_{rod}(\omega)$ yields:

$$\exp_{SE(3)}(\omega, \nu) = \begin{bmatrix} \exp_{SO3}(\omega) & \mathbf{V}\nu \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.23}$$

where $SE(3)$ is the Special euclidean Lie group of rigid body transformations in $\mathbb{R}^3$, ν is a rotated version of t and $\mathbf{V} = \mathbf{I} + \frac{(1-\cos(\alpha))}{\alpha^2} + \frac{\alpha - \sin\alpha}{\alpha^3}[\alpha]_\times^2$. In [177] also the exponential map for similarity transformations is given. As we will need this parametrization for loop closing in Chapter 7 it is briefly subsumed. A similarity transformation $\mathbf{S}$ adds a positive, uniform scaling parameter $s = e^\sigma$ with $s \in \mathbb{R}^+$ to the euclidean transformation. The corresponding exponential map is:

$$\mathbf{S} = \exp_{Sim(3)}(\omega, \nu, \sigma) = \begin{bmatrix} e^\sigma \exp_{SO(3)}(\omega) & e^\sigma \mathbf{V}\nu \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.24}$$

During optimization the authors of [177] use the following update scheme (Equation 2.4):

$$\mathbf{M}_{i+1} = \exp_{SE(3)}(\mathbf{dx}) \cdot \mathbf{M} \tag{2.25}$$

where $d\mathbf{x} = [\mathbf{d}\omega, \mathbf{d}\nu]^T$. An extensive introduction to the concepts can be found in [175]. The author also states why the update has to be performed in terms of multiplication instead of vector addition. But, in [73] it is also remarked that for smaller rotations represented by ω_1 and ω_2 the composite rotation is represented to first order by $\omega_1 + \omega_2$, i.e. $\mathbf{R}_{rod}(\omega_1 + \omega_2) \sim \mathbf{R}_{rod}(\omega_1)\mathbf{R}_{rod}(\omega_2)$. In fact many algorithms work with latter assumption, i.e. the update step is computed as follows:

$$\mathbf{M}_{i+1} = \begin{bmatrix} \mathbf{R}_{rod}(\omega_i + d\omega_i) & \mathbf{t}_i + d\mathbf{t}_i \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.26}$$

We performed a small simulation to test the accuracy, speed and convergence behaviour of different update rules.

2.2.1.5 Speed and Convergence

Due to the drawbacks of using unit quaternions or Euler representations mentioned above, we only evaluate the convergence rates, speed and accuracy of Rodrigues and Cayley representations. Therefore, we perform pose optimization given a fixed number of 500 map points and their corresponding projections in a virtual camera. The points are sampled in world coordinates and subsequently projected to the camera from ground truth poses. The ground truth rotation matrix $\mathbf{R}_{gt}$ is created using Equation 2.14 and the angles are sampled in a range of $[0, 2\pi]$. Then we add noise to image observations and the ground truth pose and convert them to the corresponding minimal representation. The perturbed ground truth pose is used as initial pose for the iterative refinement. For optimization we use g2o [102] with numerical gradients. Figure 2.2 depicts the results for the mean number of iterations until convergence (threshold on parameter update 1e-6), the mean execution time and the median and mean rotation and translation error. The Cayley parametrization show fast convergence and timings, but is a lot less accurate, probably due to multiple instances were the singularities were hit or the deviation from the ground truth pose were to large and convergence was not achieved. More interestingly, the multiplicative pose update using Equation 2.25 yields the same results as the additive update, but is slower (due to more operations for the translation update) and has slower convergence. Thus in the remainder of this thesis, we will use the additive update scheme for pose estimation and bundle adjustment, and the multiplicative update for the pose-pose constrains during loop closing as large differences between estimated and true pose are expected.

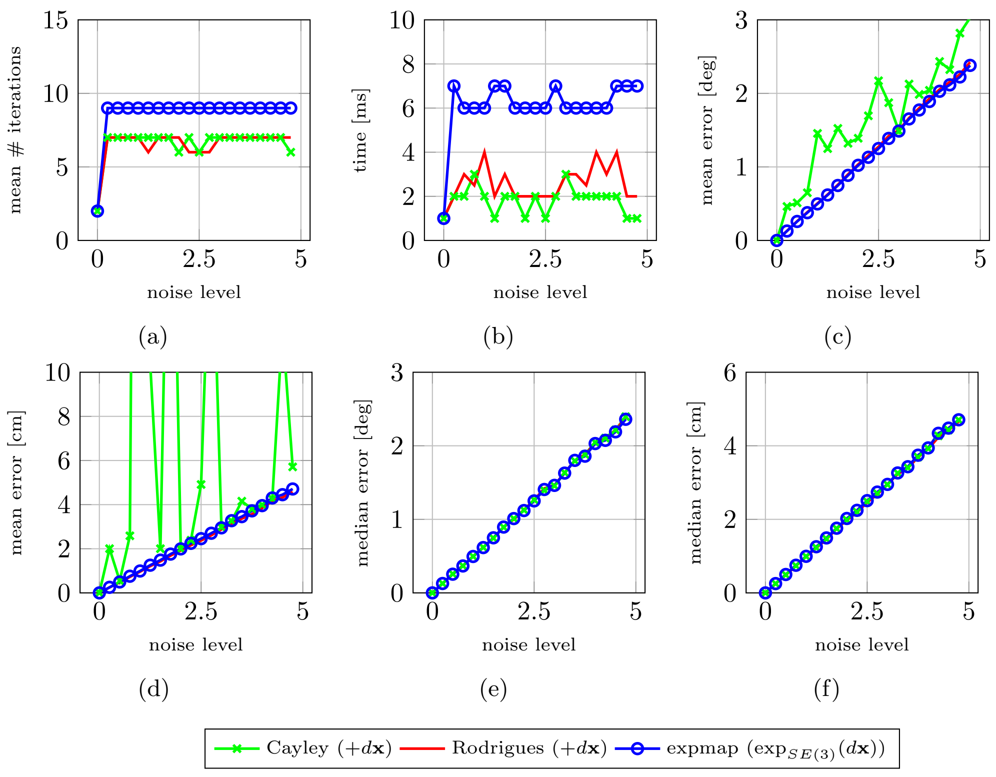


Figure 2.2. Comparison number of iterations (a), execution time (b), mean and median position error (c) and (e) and mean and median rotation error (d) and (f) of different rotation parametrizations. Rodrigues depicts the additive update rule and expmap the multiplicative.

2.3 Fisheye Cameras - Imaging Process

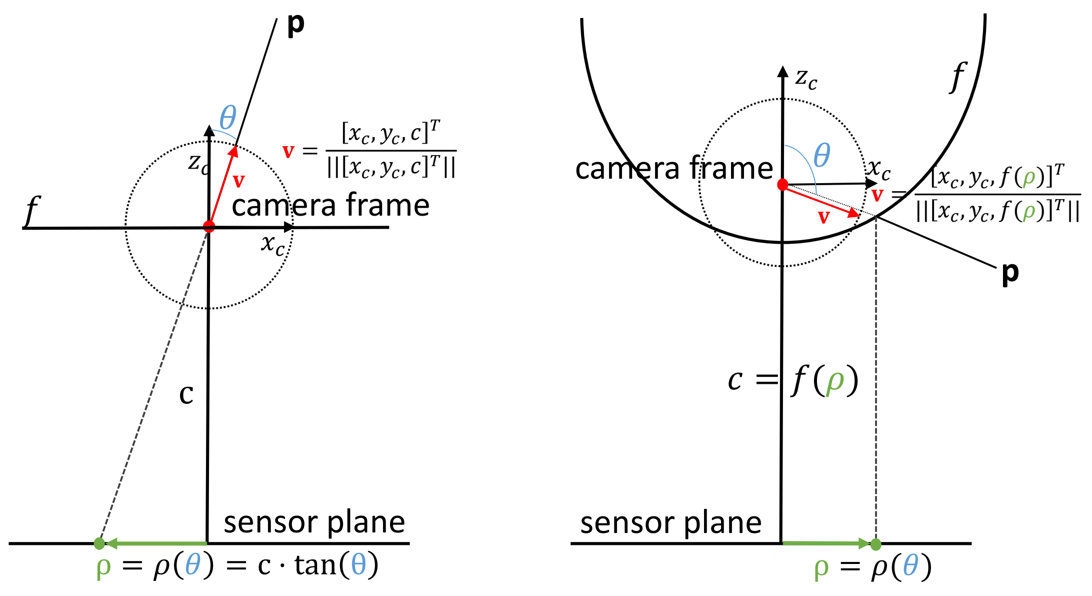
This section is a brief introduction to the basic modeling of the imaging process, i.e. of the projection of the three dimensional world to the two dimensional image plane for fisheye cameras. As fisheye cameras are used in this thesis, we focus on omnidirectional cameras and especially the camera model of [158, 159] and outline the difference to the classic perspective camera model. For a more detailed introduction to omnidirectional cameras the reader is referred to [123, 144]. The estimation of the parameters involved in the imaging process is subject of Chapter 3. A closer look w.r.t bundle adjustment and multi-camera systems is then given in Chapter 4.

Omnidirectional cameras can be grouped into dioptric and catadioptric imaging systems, that either consist of a single lens (dioptric) or of a mirror (catopric)/lens combination (catadioptric). Furthermore, an omnidirectional camera is either a central or a non-central projection system. In general, central cameras satisfy the single viewpoint property, i.e. they have a unique projection center and thus a one-to-one mapping between an image pixel and a corresponding direction vector. This mapping is depicted in Figure 2.3 for perspective and fisheye cameras. The single viewpoint property is often desirable, as the well-known theory of epipolar geometry ([73]) can be easily established and in addition the image can be mapped to a perspective projection.

Thus a perspective camera is an example of a central projection system, where all rays from world points meet in the projection center and this pencil of rays intersects the image plane. This representation, however, is restricted to incident angles θ smaller than 90 ° from the principal axis, i.e. the axis that is perpendicular to the sensor plane. This observation can also be made from the central projection equation:

$$\rho = \rho(\theta) = c\tan(\theta) \tag{2.27}$$

where ρ is the radial distance of an image point from the principal point, c the focal length, i.e. the perpendicular distance of the camera coordinate system from the sensor plane and θ the incident angle of a ray. This projection is depicted in Figure 2.3a. Equation 2.27 represents the back-projection of a world point to the image plane. The forward projection, i.e. the mapping of an


p
θ
$\mathbf{v} = \frac{[x_c, y_c, c]^T}{||[x_c, y_c, c]^T||}$
z_c
v
f
camera frame
x_c
c
sensor plane
$\rho = \rho(\theta) = c \cdot \tan(\theta)$
f
z_c
camera frame
θ
x_c
$\mathbf{v} = \frac{[x_c, y_c, f(\rho)]^T}{||[x_c, y_c, f(\rho)]^T||}$
v
p
$c = f(\rho)$
sensor plane
$\rho = \rho(\theta)$

(a) perspective projection (b) general projection

Figure 2.3. Shown are two different types of imaging functions. (a) depicts the classic perspective projection. The radial distance function $\rho(\theta)$ projects to the sensor plane scaled by the focal length c. The forward projection from sensor plane to the direction vector $\mathbf{v}$ is simply given by normalizing the vector $[x_c, y_c, c]^T$ to unit length. The angle θ is restricted to $\theta > -\pi/2$ and $\theta < \pi/2$; (b) depicts a more general projection. The forward and backward projection are described by a polynomial $f(\rho)$ and $\rho(\theta)$ respectively. Here incident angles $> \pi/2$ and $< -\pi/2$ are possible. In this model the points are first projected to the camera frame and subsequently to the sensor plane with an orthographic projection.

image point to its bearing vector in the camera frame is simply given by setting the z_c coordinate to the focal length c and normalization to unit length:

$$\mathbf{v} = \frac{[x_c, y_c, c]^T}{||[x_c, y_c, c]^T||} \tag{2.28}$$

In this work, fisheye lenses with incident angles greater 90° are used. Thus, a projection that extends the limited range of the central projection in Equation 2.27 is needed. For fisheye lenses the projection $\rho(\theta)$ usually depends on the manufacturing process, where various projections such as equisolid, equidistant or stereographic projections can be found. In [162] various such functions are studied and calibrated. Instead of expressing the back projection with a fixed function, [159] model it with a polynomial of degree n:

$$\rho = \rho(\theta) = b_0 + b_1\theta + b_2\theta^2 + ... + a_n\theta^n \tag{2.29}$$

The bearing vector is then given by setting the z' coordinate to $f(\rho)$, i.e. the forward projection function:

$$\mathbf{v} = \frac{[x_c, y_c, f(\rho)]^T}{||[x_c, y_c, f(\rho)]^T||} \tag{2.30}$$

In this section we focused on the projection view of the imaging process. Converting these representations to explicit image point coordinates is straightforward and will be formulated in subsequent chapters (see Chapter 3 and Chapter 4).

2.4 Computer Graphics

This section will give an introduction to the basic concepts of OpenGL (Open Graphics Library) as a tool for rendering 3D-models. The OpenGL specifications were developed by Silicon Graphics in

1994 and are now managed by the non-profit consortium Khronos Group. Today several rendering libraries exist, e.g. DirectX (Windows only), Metal (OS X only), and Vulkan (next generation OpenGL). Vulkan is the new API annouced by the Khronos Group in 2015. Vulkan is supposed to introduce less computational overhead on the CPU and distribute work in multi-threaded applications. The performance increase, however, comes at the cost of implementation complexity. Thus, in this thesis OpenGL will be used because it is open, can be used across platforms and rather easy to use.

At its core, OpenGL specifies inputs and outputs for functions that are used to create 2D and 3D computer graphics and as such to convert geometric primitives into pixels, a process known as rendering. In this thesis we will be using OpenGL specifications starting from version 3.3 (also called core-profiles) instead of the older fixed-function pipeline.

In computer graphics, a 3D point is called vertex and a connection of multiple vertexes a face. In modern OpenGL, the complete rendering pipeline can be divided into multiple fixed and programmable steps, whereas only two are used in this thesis, i.e. the vertex and the fragment shader. Shaders in turn are small programs that can be executed in parallel exploiting the fact that modern GPUs (graphics processing units) consist of a large number of small programmable processors (shader cores). In OpenGL, shaders are written in a C-like language called GLSL. In this thesis the vertex shader will be used to render a geometrically correct fisheye projection and the fragment shader, to color the pixels according to their normal direction or assign a depth to each pixel.

Vertex shader

The vertex shader takes pose matrices, vertexes, normals etc. and allows to transform them arbitrarily. Usually, the transformation consists of transforming the vertex (and normal) to the camera frame followed by perspective division or orthographic projection. As we want to render a geometrically correct fisheye view of the scene, we can exploit the projection function from the last section (Equation 2.29). First, the vertex is projected to the camera frame using Equation 2.9. Then the incident angle θ is calculated as $\theta = atan(z_c/sqrt(x_c^2 + y_c^2))$. Subsequently the radial distance $\rho(\theta)$ from the principal point is computed. Further the offset from the image center to the principal point is added, and the point in the image plane is distorted by a affine transformation. The full mapping will be calibrated in the next chapter (see Section 3.4). Note, that the depth of each vertex (z_c) is not touched during vertex processing. Otherwise the depth-testing would not work correctly. Now the 3D geometry is projected to 2D, and can be passed on to the fragment shader.

Fragment shader

The fragment shader takes care of texturing, coloring of pixels etc. In general, the pixels can store any numerical information that can later be accessed from the rendered image. It could, for example be used to store the interpolated 3D coordinates of each transformed vertex in a 3-channel image. We use the shader mainly for two purposes, i.e. rendering a depth and a normal image of the 3D model. Later, these images are used to extract edges, that can be compared to images of the real world.

As each normal $\mathbf{n}$ has three components $\mathbf{n} = [n_x, n_y, n_z]$, we can simply create a 3-channel RGB image texture and store the normal values in the corresponding channel. In practice we set red=n_x, green=n_y and blue=n_z. The final color is then scaled between 0.0 and 1.0 by multiplying the color vector with 0.5 and adding 0.5.

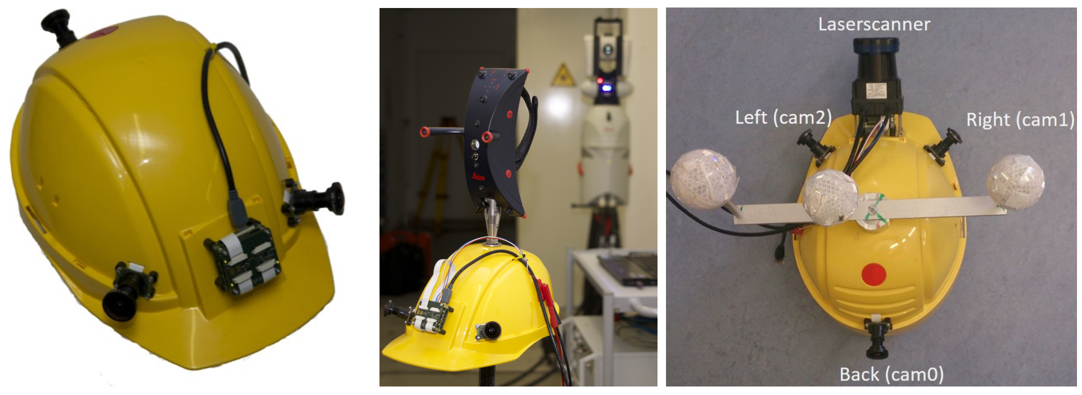
For the depth image, we color the pixels according to their depth value:

$$g = \frac{z_c - z_{min}}{z_{max} - z_{min}} \tag{2.31}$$

The bounds z_{min} and z_{max} are used to scale the depth image. The values have to be chosen according to the expected scene depth.

2.5 Datasets and Multi-Camera System

In this section, the datasets consisting of images of a multi-fisheye camera system and 6 Degrees of Freedom (DoF) ground-truth is described. We start by describing the camera system followed by a brief introduction of two different ground-truth datasets, one recorded with a lasertracker and the other with a motion capture system.



(a) (b) (c)

Figure 2.4. (a) multi-fisheye camera system attached to a helmet. On the front of the helmet, the VRmagic USB 2.0 platform is visible. (b) T-Probe attached to the helmet. The lasertracker is visible in the background of the image. (c) Helmet system with rigid body that consists of three reflective spheres.

Name	GT	#frames	length [m]	dur [s]	avg m/s	avg °/s
Basement 1 forth/back	laser	1019	18.8	40.76	0.46	13.64
Basement 2 fast	laser	680	18.6	27.60	0.67	12.15
Indoor 1 Static	mocap	1017	15.3	56.30	0.27	24.04
Indoor 2 Dynamic	mocap	759	19.7	63.24	0.64	25.28
Outdoor 1 Dynamic	mocap	1044	21.01	65.45	0.32	27.23
Loop 1 Basement	-	6339	97.76	253.56	0.39	13.62
Loop 2 Floor/Rooms	-	6256	69.33	250.24	0.27	14.55

Table 2.1. Trajectory statistics. Depicted are the number of frames, the length in meter, the duration in seconds and the average translation and rotation velocity. If ground truth is available it is either recorded with the lasertracker (laser) or the motion capture system (mocap).

Multi-Fisheye Camera System

The MCS consists of a multi-sensor USB 2.0 platform from VRmagic (model VRmC-12) with integrated Field Programmable Gate Array (FPGA) and is depicted in Figure 2.4a. Hardware triggered image acquisition is handled by the platform for up to 4 imaging sensors. However using the maximum number of sensors decreases the possible frame rate which is bounded by the USB 2.0 transfer rate. Thus, we use 3 CMOS sensors with a resolution of 754×480 pixels, allowing the system to run at 25Hz without dropping frames. The sensors are equipped with fisheye lenses (Lensagon BF2M12520) with 185° FoV and are mounted to a helmet. The image have a small overlap due to the large FoV.

Lasertracker Dataset

This dataset was recorded in a basement corridor. The lasertracker is a Leica AT-901 that we use together with a Leica T-Probe. It measures the distance to the T-Probe with an absolute interferometer and determines the vertical and horizontal angles to the target. The position of the T-Probe is calculated by polar point determination and the rotation angles are estimated with an integrated telecentric camera, that extracts infrared markers on the T-Probe. The standard deviation of the position is about $\sigma_{pos} \approx 100\mu m = 0.1mm$ and for the rotational component about $\sigma_{angle} \approx 0.05mrad$. We use the internal hardware trigger signal of the MCS to induce a 6DoF measurement of the T-Probe which runs at a higher rate than the MCS. The assembly is depicted in Figure 2.4b.

Motion Capture Dataset

This dataset was recored indoor and outdoor with dynamic and static scenes, i.e. with and without moving objects. We use an OptiTrack motion capture system with eight hardware triggered high-speed cameras (Prime 17W). The system is calibrated in advance by waving a calibration stick consisting of three passive retro-reflective markers in the volume that the cameras observe. The camera poses are recovered metrically, as the dimensions of the calibration stick are known.

Once the system is calibrated, the 3DoF position of passive markers can be tracked by triangulation with sub-millimeter accuracy with 360Hz. The 6DoF pose of a rigid body (multiple rigidly coupled markers) can be determined by observing at least three markers and fitting a coordinate frame to them. The rigid body is depicted in Figure 2.4c.

Trajectory and Scene Characteristics

To test and evaluate the methods developed in this thesis, we recorded multiple trajectories with different characteristics. We recorded dynamic as well as static scenes with different translational and rotational velocities, to evaluate the robustness of the developed motion estimation methods. In addition, outdoor as well as indoor scenes, are recorded covering narrow and wide areas as well as different illumination conditions. The trajectory characteristics are subsumed in Table 2.1.

2.6 Summary

In this chapter, an overview of several important models, principles and tools that are frequently used in this thesis are given. First the essentials of (robust) iterative non-linear least-squares techniques were recapitulated. Then, the camera pose representation was introduced, followed by different versions of minimal rotation parametrization. Subsequently, we outlined the employed fisheye projection and showed the difference to the classic central projection. Finally, the OpenGL rendering pipeline used in this thesis was introduced.

3. Improved Omnidirectional Camera Calibration

In this chapter, an improved version of the widespread calibration procedure of [158] and its online available implementation [38] is presented. Most of this chapter is published in the journal article:

> S. Urban, J. Leitloff, and S. Hinz. Improved Wide-Angle, Fisheye and Omnidirectional Camera Calibration. *ISPRS Journal of Photogrammetry and Remote Sensing*, 108:72–79, 2015.

The camera model was first proposed in [158]. By modeling the imaging process with a polynomial, it is general and does not require prior knowledge about the projection process. This is especially useful for the case of multi-camera rigs, that could consist of different cameras, e.g. perspective, fisheye or catadioptric. Instead of having to provide different projection equations and Jacobian, each camera can be modeled with a unified model. The camera model is used throughout this dissertation and will be applied to various problems such as multi-camera bundle adjustment (Chapter 4), distorted binary descriptors (Chapter 5.2) and rendering of fisheye images (Chapter 6). Thus, a robust and accurate calibration of all involved camera parameters is essential.

We analyzed the two-step refinement of interior and exterior orientation proposed in [159] and replaced the employed residual function in order to be able to jointly refine all parameters. Thereby we obtain better convergence behavior of the non-linear least squares optimization and it shows that the original five step calibration procedure can be conducted in only three steps. In addition, we changed the original implementation to a consistent use of subpixel accurate point measurements. An implementation of the methodology described in this chapter is available online[1] and can be used as an add-on to the original toolbox at hand.

Previous work on improving [158, 159] was conducted by [56]. They extended the camera model by replacing the affine transformation between image and sensor plane coordinates with a perspective transformation and also proposed a new method to estimate the center of distortion. They indicate that a bundle adjustment was performed in the end without providing details. Then they conducted calibrations on several datasets and compared the results to the original implementation. They observed slight improvements of the Mean Squared Reprojection Error (MSRE) (in pixel) but did not discuss why the standard deviation of the reprojection error was partly larger than the MSRE and even larger than the standard deviation of the original implementation. Further, only the reprojection error was investigated to assess the calibration quality. In addition, in this chapter, we examine the quality of the estimated calibration parameters, i.e. the covariances of interior and exterior orientation.

In the remainder of this chapter, we first recapitulate the camera model as well as the calibration procedure as proposed in [158]. Then, we detail our contributions to the existing approach and

[1] https://github.com/urbste/ImprovedOcamCalib

perform a comprehensive evaluation using datasets provided by the authors [38] as well as own datasets and show that we achieve a significant accuracy increase. In addition the improved results are compared to a different, accurate omnidirectional camera model ([122]) to substantiate the state-of-the-art performance of the proposed procedure.

3.1 State-of-the-Art

Wide-angle, omnidirectional and fisheye cameras are nowadays popular in many fields such as computer vision, robotics and photogrammetry. The large FoV is beneficial for various tasks such as navigation, localization, tracking, mapping and so on. As soon as metric information needs to be extracted, camera calibration is the first step necessary to determine the relation between a 3D ray and its corresponding mapping on the image plane.

In general, the calibration of a camera consists of two steps. The first involves the choice of an appropriate model that describes the behavior of the imaging system. In recent years, various such models for dioptric (fisheye) and catadioptric cameras have been proposed [64, 81, 122, 123, 158, 159]. In [145] a comprehensive overview is given. Conventional dioptric and catadioptric camera models rely on the single viewpoint constraint [64] that only applies approximately in practice, due to the assembly of catadioptric cameras or the lens system of dioptric cameras. Hence, recent work [166] focuses on adapting computationally demanding non-central models to the quasi-central case.

The second step is the estimation of all parameters that a certain model incorporates, i.e. interior and exterior orientation of the system as well as distortion parameters that model the differences from an ideal imaging process. Several methods for calibration have been proposed, which can be classified into three categories [216] starting with the most general one.

The first is auto- or self-calibration, where epipolar-constraints and point correspondences between multiple views are used [50, 123, 124]. In [124] the auto-calibration for fisheye and catadioptric systems is based on the robust establishment of the epipolar geometry from few correspondences in two images by solving a polynomial eigenvalue problem. Using this model coupled with Random Sample Consensus (RANSAC) outliers are removed and no prior knowledge about the scene is necessary. Subsequently, bundle block adjustment can be employed, to estimate all camera parameters and reconstruct the scene except for a similarity transformation.

The second is space resection of one image from a non-planar object with known 3D world coordinates [144, 162]. In [162] the validity of geometric projections for fisheye lenses is investigated. The camera is calibrated using one image of a room prepared with 3D-passpoints that cover the field of view of the fisheye lens. To enhance the model quality and compensate for real world deviations from the geometric model, radial as well as tangential distortion parameters are subsequently added to the projection model. A camera calibration approach for catadioptric cameras using space resection is proposed in [144]. Lifted coordinates are used to linearly project scene points to the image plane. After the projection matrix is estimated from scene points distributed over a 3D-calibration object, a final non-linear refinement is applied.

The third calibration method involves the observation of a planar object (e.g. checkerboard) with known 3D object coordinates [122, 158, 216] where at least two images of the planar object are necessary [116, 216]. The calibration procedure of [122] involves the manual selection of mirror border and center, respectively. For fisheye lenses the image center is used. The generalized focal length, which is part of the exact projection model, is estimated from at least three points on a line. To automatically extract all checkerboard corners, a planar homography is estimated from four manually selected points. Then the remaining checkerboard points are projected to their estimated location and a local subpixel corner search is performed. The initially estimated homography is based on start values such as the generalized focal length and the center of distortion and thus already imposes model assumptions. If the initial assumptions are highly biased the subsequent procedure can hardly recover. The projection model of [122] include tangential distortion parameters, whereas the polynomial approach of [158] only models radial symmetric effects. Latter initially extracts all checkerboard corners using the algorithm of [155]. The calibration procedure is based on multiple observations of a planar calibration object [216], is fully automatic and does not require further user interaction or prior knowledge. A potential disadvantage are wrong point extractions, that can bias the calibration result. Also the two-step refinement of interior and exterior orientation is critical, as all parameters are strongly correlated. Our proposed approach tries to avoid both possible disadvantages, but does not change the underlying projection model.

3.2 Camera Model

Let $\mathbf{m}' = [u', v']^T$ be a point on the image plane, i.e. the coordinates are measured from the upper left corner of the image. Its corresponding point on the sensor plane, is denoted by $\mathbf{m} = [u, v]^T$. Those two points, are related by an affine transformation $\mathbf{m}' = \mathbf{A}\mathbf{m} + \mathbf{O}_c$:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} o_u \\ o_v \end{bmatrix} \tag{3.1}$$

The transformation matrix $\mathbf{A}$ accounts for small misalignments between sensor and lens axes and is usually close to the identity. The translation $\mathbf{O}_c = [o_u, o_v]^T$ relates all coordinates to the center of distortion. Now let $\mathbf{p}_c = [x_c, y_c, z_c]^T$ be a scene point in the camera frame. Then, the following forward mapping from a 2D image point to its corresponding 3D ray is described by the imaging function π_g^{-1}:

$$\lambda \pi_g^{-1}(\mathbf{m}) = \lambda(u, v, f(u, v))^T = \lambda(u, v, f(\rho))^T = \mathbf{p}_c \tag{3.2}$$

with $\lambda > 0$ and $\rho = \sqrt{u^2 + v^2}$ being the radial euclidean distance from the image center. The subscript g stands for general. Thus, the mapping π_g^{-1} is assumed to be rotationally symmetric. For perspective cameras the mapping equals the inverse of the camera matrix $\mathbf{K}$. Function f usually depends on the mirror or lens of the system and models the projection. In literature and practical applications various projection models can be found (see Section 3.1). Instead of modeling the projection with a specific mapping function, [158] chose to approximate it with a Taylor expansion making it applicable for various lenses without prior knowledge:

$$f(\rho) = a_0 + a_2\rho^2 + ... + a_n\rho^n \tag{3.3}$$

As we want the z-axis of the camera frame to intersect with the extremum of f we assume:

$$\frac{df}{d\rho}|_{\rho=0} = 0 \tag{3.4}$$

and thus the coefficient a_1 are omitted. In the next section, the calibration procedure proposed by [159] is recapitulated briefly, which is aimed at an estimation of the mapping function π_g^{-1}. Subsequently the contributions are introduced that help to determine the parameters involved in π_g^{-1}.

3.3 Original Calibration Methodology

For this particular camera model, the goal of calibration is to estimate the parameters of the imaging function g, i.e. the $N-1$ coefficients $a_0, a_2, ..., a_{N-1}$ that describe the shape of the projection function f, as well as the parameters of the affine transformation $\mathbf{A}$ and the center of distortion $\mathbf{o}_c$. For the first calibration steps, $\mathbf{A}$ is set to be an identity matrix $\mathbf{I}_{2,2}$. The initial estimate of the center of distortion $\mathbf{o}_c$ is always set to the image center. To calibrate the camera, a planar calibration pattern ($z_i = 0$) is observed $j = 1, .., J$ times from an unknown position $\hat{\mathbf{M}}_j = [\hat{\mathbf{r}}_1, \hat{\mathbf{r}}_2, \hat{\mathbf{r}}_3, \hat{\mathbf{t}}]_j$. The $i = 1, .., I$ object coordinates are denoted by $\mathbf{p}_i = \left[x_i, y_i, 0, 1\right]^T$. The corresponding sensor plane measurements $\mathbf{m}_{ij} = [u'_{ij} - o_u, v'_{ij} - o_v]^T$ are extracted in each image. This leads to the following relation:

$$\lambda_{ij}\pi_g^{-1}(\mathbf{m})_{ij} = \lambda_{ij} \begin{bmatrix} u \\ v \\ \hat{a}_0, .., \hat{a}_{N-1}\rho^{N-1} \end{bmatrix}_{ij} \hat{=} \hat{\mathbf{M}}_j \mathbf{p}_i = \begin{bmatrix} \hat{\mathbf{r}}_1 & \hat{\mathbf{r}}_2 & \hat{\mathbf{t}} \end{bmatrix}_j \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_i \tag{3.5}$$

Note, that the third column vector $\hat{\mathbf{r}}_3$ of $\mathbf{M}$ is omitted because $z_i = 0$.

3.3.1 Linear Estimation of the Exterior and Interior Orientation

In order to solve Equation 3.5 for all interior orientation parameters, the J exterior orientations are estimated in advance. First, the scale factor λ is eliminated for each equation by vectorially multiplying both sides of Equation 3.5 with $\pi_g^{-1}(\mathbf{m})_{ij}$ yielding:

$$\begin{bmatrix} v_i(\hat{r}_{31}x_i + \hat{r}_{32}y_i + \hat{t}_3) - f(\rho_i)(\hat{r}_{21}x_i + \hat{r}_{22}y_i + \hat{t}_2) \\ f(\rho_i)(\hat{r}_{11}x_i + \hat{r}_{12}y_i + \hat{t}_1) - u_i(\hat{r}_{31}x_i + \hat{r}_{32}y_i + \hat{t}_3) \\ u_i(\hat{r}_{21}x_i + \hat{r}_{22}y_i + \hat{t}_2) - v_i(\hat{r}_{11}x_i + \hat{r}_{12}y_i + \hat{t}_1) \end{bmatrix} = 0 \tag{3.6}$$

Then, and since x_i, y_i, u_i, v_i are the known observations, the third equation which is linear in all unknowns and does not depend on f, can be used to get an initial estimate of the parameters of the exterior orientation. Subsequently, latter are substituted into Equation 3.6 and the resulting system is solved for the interior orientation parameters. At this point, a linear estimate of all involved parameters, i.e. $\hat{\mathbf{u}} = [\hat{\mathbf{R}}, \hat{\mathbf{t}}, \hat{a}_0, \hat{a}_2, ..., \hat{a}_N]^T$ is found.

3.3.2 Center of Distortion and Linear Refinement

Until now, the linear calibration procedure is carried out with the assumptions that the image center is identical to the center of distortion. To find a better estimate of the center, [159] carry out an iterative search procedure, prior to performing the non-linear refinement of all parameters. First, a regular grid is placed over the image around a given $\mathbf{o}_c$, where each grid point corresponds to a potential $\mathbf{o}_c$. Then, the linear calibration procedure is conducted for each grid point and the Sum of Squared Reprojection Errors (SSRE) is calculated. The grid point with the minimum SSRE is taken as the potential $\mathbf{o}_c$ and the process is repeated until the change between two potential $\mathbf{o}_c$ is less than 0.5 pixels ([159]). Finally, the linear estimation of interior and exterior orientation is conducted once more with the updated center point $\mathbf{o}_c$, yielding the initial start values for the two-step non-linear refinement with all parameters.

3.3.3 Two-Step Non-linear Refinement

The last step in the calibration procedure of [159] is a non-linear refinement involving all calibration parameters, i.e. the exterior orientation of each camera w.r.t the calibration pattern as well as the interior orientation parameters. Instead of jointly optimizing all parameters, [159] split the estimation in two steps, i.e. interior and exterior orientation are estimated separately. They state that the convergence is not affected but the runtime is decreased. The following residual is calculated for each point correspondence:

$$r_s(\mathbf{m}', \hat{\mathbf{m}}') = \sqrt{(u' - \hat{u}')^2 + (v' - \hat{v}')^2} \tag{3.7}$$

and the non-linear least squares problem is solved:

1. Exterior orientation
   In a first step the exterior orientation of all camera poses is optimized. The interior orientation parameters are fixed to the values obtained from the last linear estimation, thus:

   $$\min_{\hat{\mathbf{M}}} \|r_s\|^2 = \min_{\hat{\mathbf{M}}} \sum_{j=1}^{K} \sum_{i=1}^{L} r_s(\mathbf{m}'_{ij}, \hat{\mathbf{m}}'_{ij}(\hat{\mathbf{M}}, \underbrace{\mathbf{a}, \mathbf{A}, \mathbf{o}_c, \mathbf{p}}_{fixed}))^2 \tag{3.8}$$

   where $\mathbf{a} = [a_0, a_2, ..., a_N]$ are the polynomial coefficients. The affine matrix $\mathbf{A}$ is initialized with the identity matrix for the first iteration.

2. Interior orientation
   Now the exterior orientation is fixed to the values from the previous step and the interior orientation is refined, thus

   $$\min_{\hat{\mathbf{a}}, \hat{\mathbf{A}}, \hat{\mathbf{o}}_c} \|r_s\|^2 = \min_{\hat{\mathbf{a}}, \hat{\mathbf{A}}, \hat{\mathbf{O}}_c} \sum_{j=1}^{K} \sum_{i=1}^{L} r_s(\mathbf{m}'_{ij}, \hat{\mathbf{m}}'_{ij}(\underbrace{\mathbf{M}}_{fixed}, \hat{\mathbf{a}}, \hat{\mathbf{A}}, \hat{\mathbf{o}}_c, \mathbf{p}))^2 \tag{3.9}$$

   Observe also, that the affine matrix $\mathbf{A}$ is now estimated for the first time.

3.4 Analysis and Contributions

We conducted various test calibrations with different datasets and iteratively repeated the two step refinement proposed in [159], but could not reproduce a converging solution. After each iteration (i.e. step 1. and 2. combined), we calculate the Root Mean Squared Reprojection Error (RMSRE):

$$\text{RMSRE} = \sqrt{\frac{\sum_{i=1}^{n} \|(\mathbf{m}'_i - \hat{\mathbf{m}}'_i)\|^2}{2n}} \tag{3.10}$$

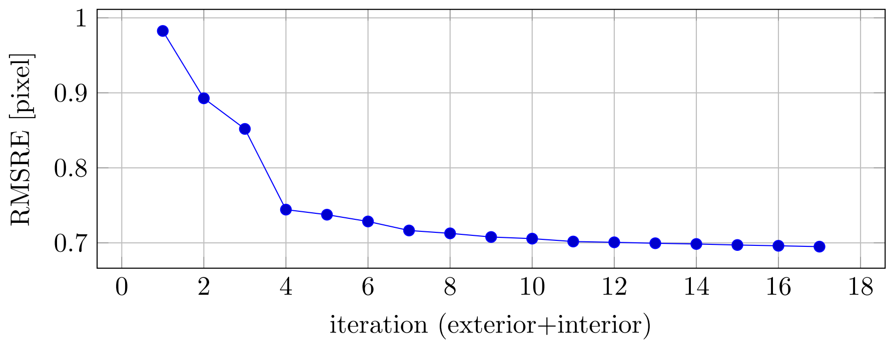


Figure 3.1. The RMSRE for iteratively and independently refining the exterior and interior orientation parameters with the original Matlab toolbox as proposed in [159]. Note that the iterations refer to iterating the refinement procedure (Equation 3.8 and Equation 3.9) and not the iterations within each refinement. Here the 'Fisheye1' data set from Section 3.5 was used.

where n is the total number of point correspondences over all images.

Figure 3.1 depicts the effect on the RMSRE by alternately refining exterior and interior orientation for the *'Fisheye1'* dataset (see Section 3.5), i.e. each iteration represents the calibration result of the two-step refinement as detailed in the previous chapter. The result of each iteration is used for the next iteration. This example indicates, that the iterative two-step procedure yielded a local minimum as the RMSRE seems to converge. However, the RMSRE after the 17-th iteration is still above our, in the following presented method (cf. Table 3.1).

As one set of the parameters (Equation 3.8: $\mathbf{A}, \mathbf{a}, \mathbf{o}_c$) is always fixed while the other set of parameters (Equation 3.8: $\hat{\mathbf{M}}$) is optimized and the RMSRE is still decreasing, it is obvious that these two sets of parameters are correlated, i.e. it is doubtful that the algorithm will converge to a correct minimum. We also observed, that the two-step refinement yielded slightly different results (e.g. in the order of pixels for the principal point) for slightly different initial distortion centers. This indicates that a global minimum is not attained.

Considering again Equation 3.8 and Equation 3.9 and the respective residual function (Equation 3.7), we can observe a potential disadvantage of the proposed error function. I points appear in J images and therefore should yield $n = 2 \cdot I \cdot J$ observations and thus rows in the Jacobian. Equation 3.7, however, results in $n = I \cdot J$ observation equations, thus each point actually accounts for two observations and Equation 3.7 incorporates both into one observation equation.

Instead of using the original two-step optimization method, we propose a procedure that jointly refines all calibration parameters. Using the image center as principal point, together with the linearly estimated parameters obtained from Section 3.3.1 as initial values is sufficient for the non-linear least squares minimization to converge quickly.

The first step is to reformulate Equation 3.7 and introduce a different residual function that fully exploits the information at each point:

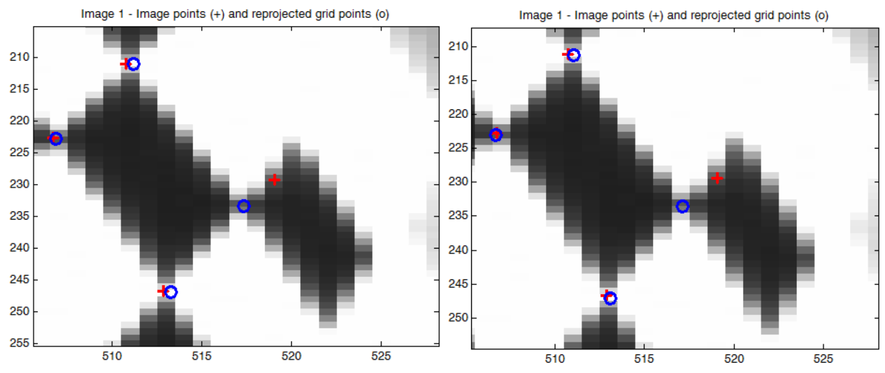
$$r_{new}(\mathbf{m}', \hat{\mathbf{m}}', k) = \mathbf{m}'_k - \hat{\mathbf{m}}'_k \tag{3.11}$$

where $k = 1, 2$ is either the u' or the v' component of the coordinate pair $\mathbf{m}'$. Now we can substitute Equation 3.11 to Equation 3.8 or Equation 3.9 respectively which now yields $n = 2 \cdot I \cdot J$ observation equations:

$$\min_{\hat{\mathbf{x}}} \|r_{new}\|^2 = \min_{\hat{\mathbf{x}}} \sum_{j=1}^{J} \sum_{i=1}^{I} \sum_{k=1}^{2} r_{new}(\mathbf{m}'_{ij}, \hat{\mathbf{m}}'_{ij}(\hat{\mathbf{x}}, \mathbf{p}), k)^2 \tag{3.12}$$

with $\hat{\mathbf{x}} = [\hat{\mathbf{M}}, \hat{\mathbf{a}}, \hat{\mathbf{A}}, \hat{\mathbf{o}}_c]$ being the parameter vector. We minimize the non-linear least squares problem Equation 3.12 using Matlab's *lsqnonlin* function with Levenberg-Marquardt regularization (cf. Equation 2.6).

At first glance, the expanded observation equations (Equation 3.8 and Equation 3.12) should be equal. The major difference lies in the residual function and the resulting normal equations. The original residual function (Equation 3.7) incorporates both observations of one image point into one



(a) bundle adjustment (b) robust bundle adjustment

Figure 3.2. Influence of falsely extracted points. The red crosses denote the extracted checkerboard points. The blue circles are the reprojected points. (a) All reprojected points are slightly biased towards the wrong checkerboard point. (b) By down-weighting that point, the reprojected points almost coincide with all extracted points and the bias is reduced.

observation equation. Hence the finite differencing step of *lsqnonlin* leads to different derivations at the point of the respective unknowns. In addition, it is not differentiable for zero residual.

Performing several calibrations with different data sets (see Section 3.5) shows, that the search for the center of distortion can be completely omitted, since a minimum is consistently found. We hereby reduce the number of necessary calibration steps from five to three, i.e. linear estimation of exterior and interior orientation and non-linear refinement of all parameters.

Furthermore, by reducing the number of subsequent calibration steps, the total runtime of the calibration decreases as well (see Table 3.4), although the runtime of the proposed one-step refinement is slightly higher than the two-step refinement, due to larger normal equations. Besides, latter pertains only if the two step refinement is carried out once. To obtain an adequate calibration with [159] the two step procedure has to be iterated multiple times leading to even longer runtime.

Usually it is not necessary to apply a robust optimization method to camera calibration. But by analyzing the extracted points from [155], we found that wrong corner points are extracted from time to time. This happens especially if the calibration pattern is observed from very oblique angles. Those images, however, are important when using planar calibration patterns to reduce the correlation of interior orientation parameters [116]. Figure 3.2 depicts an example of a falsely extracted corner point. Instead of leaving the removal of such points up to the user, we can down-weight their influence during iterative application of the non-linear optimization and treat them as outliers. We do that by applying a robust M-estimator to our proposed method and performing iterative re-weighted least squares:

$$\min_{\hat{\mathbf{x}}} \|r_{new}\|^2 = \min_{\hat{\mathbf{x}}} \sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{k=1}^{2} \omega_c({r_{new}}^{h-1}) r_{new}(\mathbf{m}'_{ij}, \hat{\mathbf{m}}'_{ij}(\hat{\mathbf{x}}, \mathbf{p}), k)^2 \tag{3.13}$$

where the superscript h indicates the h-th iteration. For the weight function ω we use Huber's function (cf. Section 2.1)

$$\omega_e(r) = \begin{cases} 1, & |r| \leq e \\ e/|r|, & |r| > e \end{cases} . \tag{3.14}$$

with $e = 1.345$ is the tuning constant (Section 2.1). For residuals below the threshold the function behaves like normal least squares. All observations with residuals larger than that pixel get re-weighted.

3.5 Results and Comparisons

In the following, we evaluate the impact of the contributions on the calibration. In order to provide a thorough evaluation, images provided by [38] containing omnidirectional and fisheye cameras as well as three own datasets consisting of fisheye and wide-angle cameras are used. We hereby

data set	standard [38]	**proposed**	**proposed robust**	mei's tool
Fisheye1	0.84	**0.28**	**0.28**	2.85
Fisheye2	0.15	**0.08**	**0.08**	1.56
GoPro	1.35	**0.54**	**0.51**	13.22
MiniOmni	0.59	**0.38**	**0.33**	1.83
VMRImage	0.39	**0.25**	**0.24**	0.27
Ladybug	0.70	**0.28**	**0.27**	0.27
KaidanOmni	0.33	**0.22**	**0.22**	1.51

Table 3.1. RMSRE in pixel for all datasets. Each dataset is calibrated with four methods, i.e. standard [38], proposed refinement, robust refinement and mei's toolbox [121]. Each method uses sub-pixel point measurements.

covered different camera models from small low-cost fisheye lenses to larger consumer cameras like the GoPro as well as omnidirectional cameras. To facilitate the comparison to all real world datasets we name them after their corresponding filename. Our own datasets are called [*'Fisheye1'*, *'Fisheye2'*, *'GoPro'*]. Data set Fisheye1 was captured with lenses from Lensation (BF2M15520) having a focal length of 1.55mm and a FoV of approx. 185° mounted on GiGE Vision cameras. Dataset *'Fisheye2'* was also captured with lenses from Lensation (BF2M12520) with a focal length of 1.25mm and a field of view of approximately 185° mounted on a USB-platform from VRmagic. The datasets provided by [38] are called [*'MiniOmni'*, *'VMRImage'*, *'Ladybug'*,*'KaidanOmni'*] and consist of three omnidirectional cameras and one wide-angle camera. Apart from the RMSRE value, we analyze the covariance matrices after the non-linear least squares estimation and extract standard deviations for the estimated parameters.

The main contribution is the reformulation of the non-linear least squares problem (Equation 3.11,Equation 3.12). The column 'proposed' in Table 3.1 depicts the RMSRE results for all datasets. Clearly the RMSRE for all datasets is significantly decreased, e.g from from 0.84 to 0.28, i.e. by factor 3 for the *'Fisheye1'* dataset. Furthermore, for all datasets that contained outlier measurements, the robust bundle adjustment yielded even better results. Overall, the mean decrease for the RMSRE over all datasets is about 2.1, proofing the validity of the proposed calibration. The overall performance gain between the standard implementation [38] and our proposed method is also graphically visualized in Figure 3.3.

As the RMSRE is only an indication of how well the estimated model describes the observations, we also assessed the quality of the estimated parameters. Since the redundancy of our proposed bundle adjustment is large, we can calculate the empirical covariance matrix $\mathbf{\Sigma}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}$. The main diagonal of $\mathbf{\Sigma}_{\hat{\mathbf{x}}\hat{\mathbf{x}}}$ contains the variance information of all unknown parameters $\hat{\mathbf{x}}$. As the standard implementation separates the refinement into two steps, we extract one covariance matrix after each step separately, i.e. yielding a covariance matrix for interior and exterior orientation. Note that the separated covariance matrices do not take the aforementioned correlations between the calibration parameters into account. This statistical fact additionally reveals the disadvantage of refining interior and exterior orientation separately. Hence, a comparison of the covariance matrices between the standard and our proposed approach should only be cautiously interpreted. Table 3.2 depicts the median standard deviation for orientation and translation averaged over all images for all methods. The standard deviation of the exterior orientation decreased 1.8 times on average. Table 3.3 depicts the standard deviation for two interior orientation parameters, i.e. first affine parameter and the first polynomial coefficient. For the sake of clarity, all other interior orientation parameters are omitted. The tendencies are the same, i.e. on average the standard deviation are 1.6 times lower.

In a last step, we studied the runtime differences of both approaches, as [158] argues that the two-step refinement reduces the runtime of the calibration procedure. We assembled a script to execute all calibration related functions without user interaction and measured the execution time. The standard five step calibration procedure was carried out with only one refinement iteration. Table 3.4 depicts the overall results. Obviously the reduction from five to three calibration steps decreased the runtime. The elimination of the center of distortion search fully compensates for the slightly longer runtime of the non-linear refinement.

dataset	standard	**proposed**	**robust**
	[mm]	[mm]	[mm]
Fisheye1	0.21	**0.09**	**0.09**
Fisheye2	0.34	**0.19**	**0.19**
GoPro	0.51	**0.39**	**0.38**
MiniOmni	0.47	**0.27**	**0.23**
VMRImage	0.27	**0.38**	**0.38**
Ladybug	0.22	**0.27**	**0.26**
KaidanOmni	0.31	**0.39**	**0.39**

(a)

standard	**proposed**	**robust**
[mrad]	[mrad]	[mrad]
2.07	**0.68**	**0.73**
0.87	**0.40**	**0.40**
1.00	**0.64**	**0.63**
3.93	**1.83**	**1.62**
3.95	**2.36**	**2.35**
3.18	**1.68**	**1.63**
2.68	**1.79**	**1.78**

(b)

Table 3.2. (a) the mean standard deviation $\bar{\sigma}_{xyz}$ in x,y and z direction of the translation vector in [mm]. (b) the mean standard deviation $\bar{\sigma}_{\phi\kappa\gamma}$ of the orientation about x,y and z axis in [mrad]. Subpixel measurements were used for all calibrations.

dataset	standard	**proposed**	**robust**
	[1e-3]	[1e-3]	[1e-3]
Fisheye1	0.35	**0.13**	**0.23**
Fisheye2	0.13	**0.08**	**0.08**
GoPro	0.20	**0.18**	**0.18**
MiniOmni	0.33	**0.26**	**0.23**
VMRImage	0.24	**0.45**	**0.45**
Ladybug	0.60	**0.30**	**0.29**
KaidanOmni	0.83	**0.72**	**0.72**

(a)

standard	**proposed**	**robust**
[1e-3]	[1e-3]	[1e-3]
1.71	**0.69**	**0.69**
0.51	**0.32**	**0.32**
0.79	**0.53**	**0.52**
6.13	**3.66**	**3.27**
3.21	**2.31**	**2.31**
2.40	**1.53**	**1.48**
8.59	**5.24**	**5.22**

(b)

Table 3.3. Depicts the median standard deviations $\bar{\sigma}$ for (a) the first affine parameter $\bar{\sigma}_c$ and (b) the polynomial coefficient $\bar{\sigma}_{a_0}$. Subpixel measurements were used for all calibrations.

dataset:		Fisheye1	Fisheye2	GoPro	MiniOmni	VMRImage	Ladybug	KaidanOmni
standard	[s]	48	68	211	77	35	28	32
proposed	[s]	**37**	**43**	**196**	**58**	**18**	**12**	**18**

Table 3.4. Runtime differences between the standard procedure and our proposed three-step procedure.

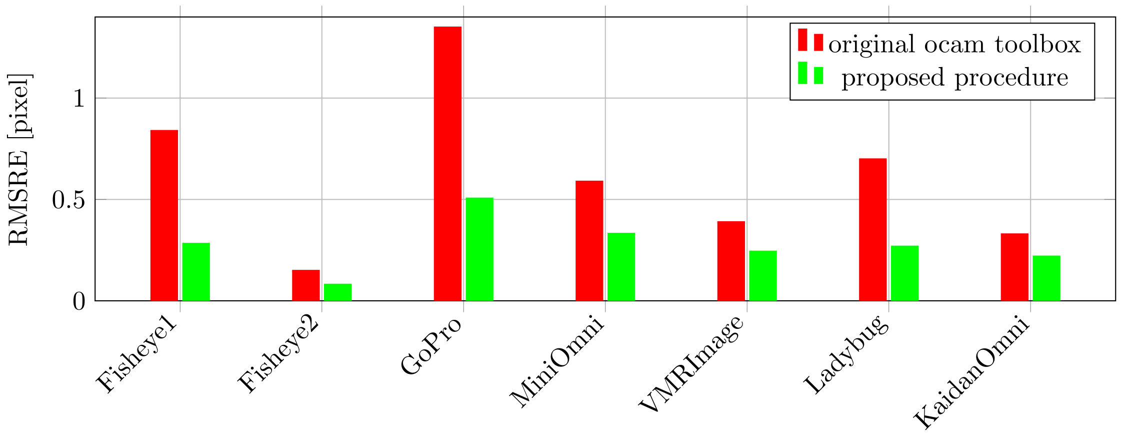
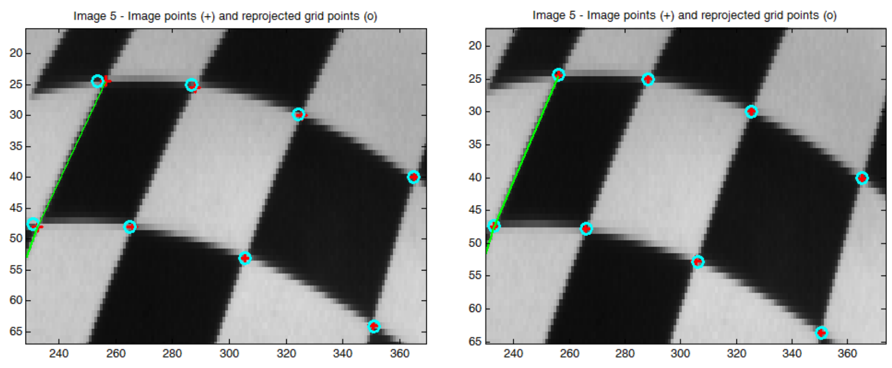


Figure 3.3. Overall performance gain for the RMSRE between the standard implementation [38] and our proposed non-linear refinement. Both use sub-pixel accurate point measurements.



(a) standard method (b) proposed method

Figure 3.4. Visual comparison of the reprojected (cyan circles) corner points (red crosses) for (a) the standard method and (b) our proposed robust refinement. This is an arbitrary region of the checkerboard from image 5 of the *'Fisheye2'* dataset (Section 3.5).

3.6 Summary

In this chapter, we extended the widely used method and implementation for fisheye and omnidirectional camera calibration of [38, 158, 159]. We reformulated the non-linear least squares refinement and applied a residual function that allows to jointly optimize all calibration parameters. We further reduced the necessary calibration steps to three and evaluated the results with various datasets including wide-angle, fisheye and omnidirectional imaging systems. It showed that a significant performance increase could be achieved. Since the automatic corner extraction yields false locations from time to time, we extended the non-linear refinement with an M-estimator to make it robust against outliers. Finally, Figure 3.4 depicts a visual comparison of the reprojected corners between the standard method on the left side and our proposed method on the right. Evidently, after a calibration with the proposed method, the reprojected points (cyan circles) coincide much better with the extracted corner points (red crosses). A Matlab implementation of all contributions is available online and can be used as an add-on to the existing toolbox [38].

4. MultiCol - Bundle Adjustment for Multi-Camera Systems

In this chapter, we present a generic, modular bundle adjustment method for pose estimation, simultaneous self-calibration and reconstruction for arbitrary MCSs. Thus, this chapter introduces the fundamentals that are at the core of all point- and edge-based mapping and navigation tasks that are approached and developed in this work. In contrast to other methods that use 3D bearing vectors (camera rays) as observation to be independent of the camera type, we extend the common collinearity equations that express all 2D image observations as a function of all unknowns. This compact formulation allows us to either calibrate the camera system, the cameras, reconstruct the observed scene and/or simply estimate the pose of the system by including the corresponding parameter block into the Jacobian matrix. In addition, we show how to express the given model in terms of a hyper graph-based representation that can be readily integrated into optimization frameworks such as g2o [102]. Apart from evaluating the implementation with comprehensive simulations, we benchmark our method against recently published methods for pose estimation and bundle adjustment for multi-camera systems in simulations and on the lasertracker datasets. Finally, all methods are evaluated using the 6DoF ground truth lasertracker datasets (see Section 2.5). Most contents of this chapter was published in the journal article:

> S. Urban, S. Wursthorn, J. Leitloff, and S. Hinz. MultiCol Bundle Adjustment: A Generic Method for Pose Estimation, Simultaneous Self-Calibration and Reconstruction for Arbitrary Multi-Camera Systems. *International Journal of Computer Vision (IJCV)*, pages 1–19, 2016.

4.1 Principles of Multi-Camera Systems and Related Work

Multi-camera systems, as developed in this work, embody the results of different categories of computer vision and photogrammetric research, i.e. camera calibration, calibration of stationary and mobile MCSs as well as bundle adjustment and pose estimation. This overview of related work concentrates on four core aspects relevant in this work, since a comprehensive review on each category would be beyond the scope of this chapter.

(I) Intrinsics

The first step to most geometric computer vision algorithms is camera calibration, since the relation between a 3D observation and its mapping onto the image plane needs to be established. If the MCS solely consists of perspective cameras, classic calibration procedures and models [18, 43, 116, 216] could be used. As soon as wide-angle, fisheye or omnidirectional cameras are included, different camera models [56, 64, 81, 122, 123, 158, 159, 166] have to be employed. An overview of related work was given in Section 3. In the remainder of this chapter, we use the polynomial camera model of [158] and calibrate all cameras using the proposed calibration procedure from Chapter 3.

(II) Extrinsics

To correctly model the transformation chain from world to image points, the extrinsics of each camera w.r.t a MCS coordinate frame need to be calibrated. This aspect can be subdivided into two subcategories, i.e. the calibration of stationary and mobile MCSs respectively.

- *Stationary*: Numerous publications deal with the calibration of stationary multi-camera systems. They use calibration objects like LED-markers, laser pointers [11, 14, 103, 181], reference bars [118] or active self-calibration [21]. Some use pairwise fundamental [11] or essential [103] matrix estimation or assume overlapping cameras in subregions of the volume [103, 181] to recover the orientation of all cameras and extract the projection matrices, usually followed by bundle adjustment [181]. Many of those methods are used to calibrate motion capturing or rigid body tracking systems.
- *Mobile*: The calibration of moving stereo platforms is well studied and implementations for pinhole camera models are available [18]. By observing a calibration pattern which is visible in both images of the stereo platform at all times, the epipolar geometry and, with it, the relative orientation can be recovered. The calibration of a MCS with at least two outward pointing cameras and possibly non-overlapping FoV is more challenging. In [108] a method is proposed that makes use of a planar calibration board with a random pattern. First, feature points with corresponding descriptors are extracted from the calibration pattern. Subsequently those descriptors are matched to images from the same pattern that is moved around the camera rig and bundle adjustment is performed to calibrate the extrinsics of the MCS. Instead of moving the calibration object about the camera system, In [100] a planar mirror is used to observe the fixed calibration object. Thus the connection between the 3D coordinates of the calibration object and the 2D observations in each camera can be established for each frame.
  In [133] a MCS with non-overlapping field of view is calibrated by extracting the mutual orientation from a common motion. Further, critical motions as well as their impact on the accuracy of the mutual orientation estimation are examined. The authors of [27] calibrate a multi-camera rig using a SLAM approach. For each camera in the rig, a map of distinctive features is created and afterward fused with a 3D similarity transformation using RANSAC [51] followed by bundle adjustment refining the feature locations as well as the relative poses of the cameras. [117] use line features to calibrate the extrinsics of a MCS. They show that arbitrary cameras can be used, by employing the unified projection model [122].

(III) Absolute pose

For a single camera, pose estimation is well studied and robust, direct minimal solutions that require only three points are readily available [90]. General solutions to the so-called Perspective-N-Point problem (PnP) are numerous [76, 95, 105, 218] including one method developed in the context of this work [196], whereas minimal solutions have been proposed, e.g. by [60, 95] that allow fast and efficient outlier rejection schemes (like RANSAC). For non-perspective cameras that have multiple viewpoints the problem is known as the Non-Perspective-N-Point problem (NPnP). In general, this can be represented by a camera that does not satisfy the Single Effective Viewpoint (SVP) constraint (e.g. catadioptric build up from perspective camera and conic mirror) or by a multi-camera system consisting of multiple cameras that satisfy the SVP constraint. The direct solutions proposed by [47, 92, 168] can all be used for multi-camera systems. In [55] and subsequent work [33] a model for pose estimation and calibration of a multi-camera system is presented. The pose estimation does not depend on overlapping views and incorporates all observations to estimate the pose of a virtual camera. The proposed calibration procedure, however, assumes overlapping views.

(IV) Reconstruction

Another possible application for MCSs is classical SfM. For single cameras as well as large sets of different cameras the problem is well defined and studied [73, 116, 189] and implementations [2, 102, 113, 182, 208] as well as results on large datasets exist, e.g. [1]. In [164], a thorough bundle adjustment for MCSs that can handle different camera models as well as scene points at infinity is presented. The former is achieved by transforming each 2D image observation to bearing vectors, the latter by spherical normalization of scene points and bearing vectors. To avoid singular covariance matrices (e.g. the 3D-bearing vectors have only 2 degrees of freedom) they reduce the number of equations employing the work of [53]. In a subsequent work [163], they expanded the bundle adjustment in order to calibrate the extrinsics of the MCS. We will elaborate the difference to our method in Section 4.3 and compare the results in Section 4.5.1 and Section 4.5.3.

Summarizing, most of the four challenges that arise when using mobile MCSs have been addressed separately or in some of the referenced methods they are partly used together. To the best of our knowledge, none of the cited methods is able to handle all challenges (I)-(IV) at once including arbitrary camera models [1]. In the remainder of this chapter, we will first elaborate the classical collinearity equations and examine possible challenges for the its use with MCSs. Then, the expanded collinearity equations are introduced and differences to other approaches are emphasized. Subsequently, various simulations are performed and the proposed method is compared to publicly available algorithms. Finally, the pose estimation and bundle adjustment capabilities are evaluated using accurate and hardware triggered ground truth trajectories.

4.2 The MultiCol Model

In this section, we introduce our approach to modeling MCSs including all necessary extrinsic and intrinsic parameters. To better clarify the theoretical relations between standard formulations of bundle adjustment methods including self-calibration we also show, the derivation of these specific formulae from our general formulation. Throughout the section, we emphasize the differences to the standard formulation in each step.

The first subsection is a recapitulation of the classical collinearity equations. In addition, we analyze the issues of using the classical formulation in the MCS case. Then, the general camera model is summarized and used to augment the collinearity equations. In a last step, we add an additional transformation and detail the differences to other approaches.

Throughout the chapter, the following indexes are used:

$$\begin{array}{ll} c = 1...C & \text{camera c in MCS} \\ t = 1...T & \text{pose of MCS at time t} \\ i = 1...I & \text{scene points} \end{array} \tag{4.1}$$

4.2.1 Classical Collinearity Equations

The classical collinearity equations are the basis for many photogrammetric and computer vision applications. They express all observations (image points $\mathbf{m}'$) as a function of the unknowns:

$$\lambda_{it}\mathbf{m}'_{it} = \pi_c(\mathbf{p}_{it}) \tag{4.2}$$

where λ is a scale factor and $\mathbf{m}'_{it} = [u', v', 1]_{it}$ are the image coordinates of the i-th scene point $\mathbf{p}_{it} = [x_{it}, y_{it}, z_{it}]^T$ that is observed at time t. π_c is a projection function that maps scene points in the camera reference frame to the image plane. In the classical case π_c is expressed in terms of the calibration matrix $\mathbf{K}_c$:

$$\pi_c = \mathbf{K}_c = \begin{bmatrix} c & cs & o_u \\ 0 & c(1+m) & o_v \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_u & c_u s & o_u \\ 0 & c_v & o_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

where c is the focal length, $(1+m)$ is a scale factor for the v axis, o_u, o_v are the coordinates of the principal point and s models the skewness of the u and v axis. The rigid homogeneous transformation from camera to world frame is given by:

$$\mathbf{M}_t = \begin{bmatrix} \mathbf{R}_t & \mathbf{t}_{0t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.4}$$

with $\mathbf{t}_{0t}$ being the projection center in world frame coordinates, and $\mathbf{R}_t$ the rotation matrix of world to camera system. Substituting the above transformation and projection matrix into Equation 4.2 the classical collinearity equations become:

$$\lambda_{it}\mathbf{m}'_{it} = \mathbf{K}_c\mathbf{M}_t^{-1}\mathbf{p}_i = \mathbf{K}_c\mathbf{R}_t^T[\mathbf{I}| - \mathbf{t}_{0t}]\mathbf{p}_i \tag{4.5}$$

[1]At this point it should be noted, that the presented method does not contain a direct, minimal solution to the non-central absolute pose problem. In fact it needs latter to initialize the estimate if no prior information about the pose is available.

To express the observations in the image plane as a function of all unknowns, we divide Equation 4.5 by the third row yielding:

$$
\begin{aligned}
u' &= c_u \cdot \frac{r_{11}(x-x_0)+r_{21}(y-y_0)+r_{31}(z-z_0)}{r_{13}(x-x_0)+r_{23}(y-y_0)+r_{33}(z-z_0)} + o_u + d_r(\rho) + d_{t_u}(\rho) \\
v' &= c_v \cdot \frac{r_{12}(x-x_0)+r_{22}(y-y_0)+r_{32}(z-z_0)}{r_{13}(x-x_0)+r_{23}(y-y_0)+r_{33}(z-z_0)} + o_v + d_r(\rho) + d_{t_v}(\rho)
\end{aligned} \tag{4.6}
$$

with $\rho = \sqrt{u^2+v^2}$ being the radial distance from the origin of the sensor coordinate system. Note that we now added radial and tangential distortion functions d_r and d_t respectively and omitted the skew factor s as well as the indices i and t to facilitate the reading. The most common characterization for both distortion effects can be found in [20]:

$$
\begin{aligned}
d_r(\rho) &= g_1\rho^2 + g_2\rho^4 + g_3\rho^6 \\
d_{t_u}(\rho) &= 2h_1uv + h_2(\rho^2+2u^2) \\
d_{t_v}(\rho) &= h_1(\rho^2+2v^2) + 2h_2uv
\end{aligned} \tag{4.7}
$$

where g_1, g_2, g_3 are the coefficients of the radial-symmetric distortion and h_1, h_2 model the tangential-asymmetric distortion. Analyzing Equation 4.6 we can identify the following possible challenges for the use in multi-camera systems:

1. *Point with incident angles greater* 90°: We can not distinguish, and hence not project, points that lie behind the cameras, given that a perspective camera model is used (central projection).
2. *Camera model*: To overcome the first issue, we could use an omnidirectional camera model and include it in Equation 4.6. If multiple cameras, e.g. fisheye and perspective cameras would be combined to a camera system, we have to provide different versions of Equation 4.6. This challenge could either be eluded by transforming image coordinates to bearing vectors (see Section 4.3) using the corresponding intrinsics of each camera or by using a general camera model that models all prevalent cameras. This work utilizes the latter approach, but see Section 4.3 for a comparison to [164] and [92] who use bearing vectors as observations and a reasoning why new challenges arise when optimizations are carried out over camera rays instead of image coordinates.
3. *Multiple cameras*: Finally, Equation 4.6 expresses solely the projection of scene points i to a camera at time t. In a MCS the projection has to be expanded by a transformation of scene point $\mathbf{p}_i$ to a MCS pose at time t and finally to a camera c within some MCS frame.

In the following, the expended collinearity equations are introduced as well as their graph-based formulation.

4.2.2 Expanded Collinearity Equations

In order to be able to utilize arbitrary cameras in the MCS, a suitable camera model is necessary. We chose to include the camera model proposed in [158, 159] that was introduced in Chapter 3.2, since it is not limited to specific cases and allows us to employ all prevalent cameras that are currently used in applied computer vision and robotics, e.g. perspective, dioptric (fisheye), as well as catadioptric cameras. In this section we will show the differences to the classical perspective camera and emphasize how this model helps to generalize the imaging process.

The forward projection of a point in the image plane to a vector that points in the direction of the scene points is given by function π_g^{-1} that was introduced with Equation 3.2. The generality comes from the fact, that a polynomial $f(\rho)$ was used to model the mapping process. For a classical perspective camera, the direction vector is obtained straightforward by setting the z component of Equation 3.2 to the cameras focal length c:

$$
\boxed{
\begin{aligned}
\text{general:}\ f(\rho) &= a_0 + a_2\rho^2 + ... + a_{N-1}\rho^{N-1} \qquad (4.8) \\
\text{perspective:}\ f(\rho) &= c \qquad (4.9)
\end{aligned}
}
$$

Applying subsequent spherical normalization yields the so-called bearing vector [91] (also camera ray [164]) that is abbreviated with $\mathbf{v}$:

$$\mathbf{v} = \frac{\pi^{-1}(\mathbf{m}')}{\|\pi^{-1}(\mathbf{m}')\|} = \mathsf{N}(\pi^{-1}(\mathbf{m}')) \tag{4.10}$$

where N expresses spherical normalization and π^{-1} the mapping function with corresponding function $f(\rho)$.

However, as we propose to maintain image plane coordinates as observations, the inverse mapping is needed, i.e. how a scene point projects to the image plane through a projection function:

$$\boxed{\begin{aligned} \text{general: } \rho(\theta) &= b_0 + b_1\theta^1 + ... + b_{M-1}\theta^{M-1} && (4.11)\\ \text{perspective: } \rho(\theta) &= c \cdot tan\theta && (4.12) \end{aligned}}$$

For the perspective case, this projection matches the central projection. At this point, also geometric fisheye projections could be employed to model the fisheye cameras used in this work as done in [162]. This however is not as general as using a polynomial, as different versions of Equation 4.12 had to be provided and the extension of the MCS is not straightforward. For the general case, the coefficients $b_i, i = 0, .., M-1$ of the polynomial have to be determined from the calibrated forward projection (Equation 4.8). By expressing the distance ρ from the image center as a function of the incident angle between a ray and the camera z-axis we write:

$$tan\theta = \rho/f(\rho) = \sqrt{x_c^2 + y_c^2}/z_c \tag{4.13}$$

By substituting Equation 4.13 and Equation 4.8 we get (see also [56]):

$$0 = a_0 - \rho/tan(\theta) + a_2\rho^2 + .. + a_{N-1}\rho^{N-1} \tag{4.14}$$

and ρ for a specific θ is found as the root of Equation 4.14.

It would render our approach inefficient to invert Equation 4.14 for each incident angle. Thus, an analytical version of the inverse projection $\rho(\theta)$ that is valid over the whole image is desirable. This can be achieved by calculating the inverse for a fixed range of sampled incident angles θ, e.g. from 0 to $\pi/2$ in 0.01rad steps ([38]). Then, a polynomial can be fitted to the resulting data points. To get a certain accuracy of the approximation, the degree M of the polynomial b_{M-1} is increased until the RMS of the fitted polynomial is below some threshold.

Now, let a pixel on the sensor plane:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \rho(\theta)\cos(\phi) \\ \rho(\theta)\sin(\phi) \end{bmatrix} = \frac{\rho(\theta)}{\sqrt{x_c^2 + x_c^2}} \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{4.15}$$

where ϕ is the azimuth angle of the incident ray. Finally, we can apply the affine transformation (Equation 3.1) and compare the object to image plane transformations:

$$\boxed{\begin{aligned} \text{general: } \mathbf{m}'_{it} = \pi_{g_c}(\mathbf{p}_{it}) &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & 1 \end{bmatrix} \begin{bmatrix} \rho_c(\theta)x/r \\ \rho_c(\theta)y/r \end{bmatrix} + \begin{bmatrix} o_u \\ o_v \end{bmatrix} && (4.16)\\ \text{perspective: } \mathbf{m}'_{it} = \pi_{p_c}(\mathbf{p}_{it}) &= \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_u x/z \\ c_v y/z \end{bmatrix} + \begin{bmatrix} o_u + d_{r_u} \\ o_v + d_{r_v} \end{bmatrix} && (4.17) \end{aligned}}$$

with $\mathbf{p}_{it} = \mathbf{M}_t^{-1}\mathbf{p}_i = \mathbf{p}_c$ and $r = \sqrt{x_{it}^2 + y_{it}^2} = \sqrt{x_c^2 + y_c^2}$. Subscripts g and p denote general and perspective respectively. Again, we omitted the indexes i and t to simplify the reading. In the perspective case, we exploited Equation 4.13 to eliminate $\tan\theta$ from the equation. The tangential distortion is omitted in both cases.

At first glance, the general case is missing the radial distortion. But if Equation 4.16 is extended, one can observe that the final sum is similar to the radial distortion function of the perspective case introduced in Equation 4.7. Hence, the radial distortion is included in the polynomial $\rho_c(\theta)$.

Until now, we added a general camera model to the collinearity equations. The mapping of Equation 4.16 can either be used to estimate the pose of a camera $\mathbf{M}_t$, refine the intrinsics or conduct

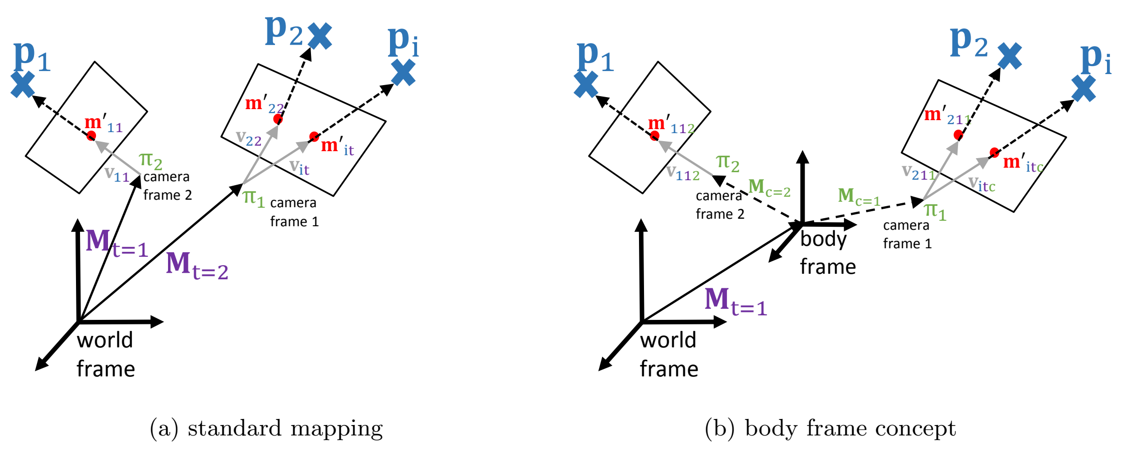


Figure 4.1. Visualizes the relation of all involved parameters. (a) depicts Equation 4.16 (b) depicts the expanded collinearity equations Equation 4.18 and the body frame concept is included.

a bundle adjustment comprising perspective as well as omnidirectional and fisheye cameras, given the image observations $\mathbf{m}'_{it}$. Thus, Equation 4.16 maps the relation of observed image points onto all unknowns for one camera at a time and leads to the well known observation equations. This mapping is depicted in Figure 4.1a. Here one camera observes the scene at two times $t = 1, 2$.

In the case of a MCS, however, multiple cameras $c = 1..C$ observe the scene $\mathbf{p}_i$ at a time t and hence all observations of all cameras at that time have to be embedded to the observation equations. Instead of transforming the scene points directly from the world frame into the camera frame, the body frame, i.e. an intermediate frame that represents the absolute pose of the MCS is used. The body frame concept is visualized in Figure 4.1b. Now two cameras $c = 1, 2$ observe the scene at one time $t = 1$. This transformation sequence is equal to [92] and [164], thus the comparison in Section 4.3 and Section 4.5 will be straightforward. The intermediate body frame allows to separate the observations from each camera whilst simultaneously combine all observations to one observation equation. Finally the extended collinearity equations, that we call **MultiCol** in the following, are:

$$\boxed{\mathbf{m}'_{itc} = \pi_{g_c}(\mathbf{p}_{itc}) = \pi_{g_c}(\mathbf{M}_c^{-1}\mathbf{M}_t^{-1}\mathbf{p}_i)} \tag{4.18}$$

where $\mathbf{M}_c$ is the rigid 4×4 transformation from body to camera frame and here both $\mathbf{M}_c$ and $\mathbf{M}_t$ are expressed in camera to world transformations, i.e. the translation is given in world coordinates.

Since the polynomial $\rho(\theta)$ is available analytically, Equation 4.18 can be linearized and used to estimate all involved unknowns, i.e. the camera pose, the intrinsics of all cameras, the scene points and the system calibration.

4.2.3 Hyper Graph Modeling

Speaking in terms of bundle adjustment the right part of Equation 4.18 can be divided into four parameter blocks $\mathbf{x} = (\mathbf{x}_i, \mathbf{x}_t, \mathbf{x}_c, \mathbf{x}_{\pi_c})$. $\mathbf{x}_i$ has 3DoF and is the parameter block for the scene points $\mathbf{p}_i$. $\mathbf{x}_t, \mathbf{x}_c$ both have 6DoF and represent the MCS pose and camera to body frame transformations respectively for which a suitable rotation parametrization has to be chosen (Section 2.2). $\mathbf{x}_{\pi_c}$ is the parameter block for the interior orientation of each camera and has varying DoF, that depend on the degree of the polynomial. Each of those blocks gives rise to a constraint on the measured image point $\mathbf{m}'_{itc}$.

Now lets define a reprojected image point $\hat{\mathbf{m}}'_{itc}$. Assuming perfectly estimated parameters, all constraints would be fulfilled and the reprojected point $\hat{\mathbf{m}}'_{itc}$ would match the measured image point $\mathbf{m}'_{itc}$. In general this is not true and we get a residual in the image plane:

$$\mathbf{r}_{itc}(\mathbf{x}) = \mathbf{m}'_{itc}(\mathbf{x}) - \hat{\mathbf{m}}'_{itc}(\mathbf{x}) \tag{4.19}$$

that depends on the chosen subset from the parameter vector $\mathbf{x}$. Expressing Equation 4.18 w.r.t to the parameter blocks we get:

$$\boxed{\mathbf{m}'_{itc}(\mathbf{x}) = \pi_{g_c}(\mathbf{x}_{\pi_c})(\mathbf{M}_c^{-1}(\mathbf{x}_c)\mathbf{M}_t^{-1}(\mathbf{x}_t)\mathbf{p}_i(\mathbf{x}_i))} \tag{4.20}$$

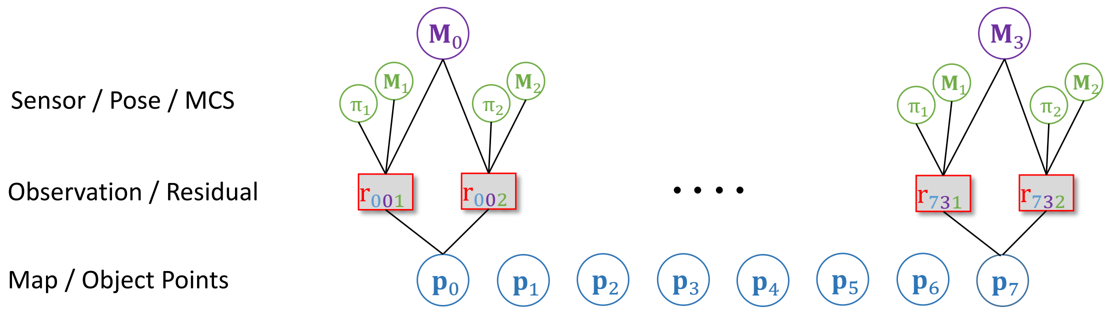


Figure 4.2. Depicted is the hyper graph modeling of MultiCol. Parameters are denoted as circles (vertices) and measurements as boxes (edges). In such a hyper graph edges can be connected to multiple vertices. In this example the $i = 1, .., 7$ map points p_i are observed by a MCS at a particular time from pose M_t, with $t = 1..3$. The MCS consists of c=1..2 cameras that have a relative orientation M_c w.r.t the body frame and an interior orientation π_c.

	image points	bearing vectors
observation	$\mathbf{m}'$	$\mathbf{v}$
dimensionality	2	3
degrees of freedom	2	2
covariance matrix	nonsingular	singular
use of intrinsics	in each iteration	once
error model	Euclidean distance	e.g. cross product, dot product [91], angle, norm of vectors difference, reduced coordinates in tangent space [164][53]

Table 4.1. Comparison of types of observations.

These constrains can be embedded in a hyper graph that can be easily integrated in a least squares graph optimization tool such as g2o [102]. A hyper graph is a graph where each edge (a residual, Equation 4.19) can be connected to multiple vertices (constraints). Figure 4.2 depicts the hyper graph that corresponds to MultiCol (Equation 4.18). The edges are represented as red boxes and each edge is connected to four vertices, that correspond to the parameter blocks in $\mathbf{x}$. During bundle block adjustment these parameter blocks can be fixed. Fixing a parameter block means that the corresponding block is removed from the Jacobian.

4.3 Difference to Other Models

The last two sections detailed the expansion of the classical collinearity equations to arbitrary rigidly coupled multi-camera systems. Our formulation uses 2D image points as observations, as all involved parameters can be expressed in terms of these image measurements. In this section, we will compare our formulation to other methods.

Basically two representations of a world point $\mathbf{p}_i$ in the camera exist. One is the projection of the scene point to the image plane (depicted as $\mathbf{m}'_{itc}$ in Figure 4.1b). This mapping is given by Equation 4.18. Here all unknowns are expressed in terms of observed 2D image points.

The second is the spherically normalized 3D direction from the camera to the corresponding scene point (bearing vector) and is given by Equation 4.10. These vectors are depicted as $\mathbf{v}_{itc}$ in Figure 4.1b). Here the interior orientation of the camera is used once, to transform the observed image point to its corresponding direction in the camera frame. Then, the reprojected bearing vector is:

$$\boxed{\hat{\mathbf{v}}_{itc} = \mathsf{N}(\mathbf{M}_c^{-1}\mathbf{M}_t^{-1}\mathbf{p}_i) = \mathsf{N}(\mathbf{p}_{itc})} \tag{4.21}$$

Now, again a residual function similar to Equation 4.19 has to be defined. Computing the distance between $\hat{\mathbf{v}}_{itc}$ and $\mathbf{v}_{itc}$ directly would yield three rows in the corresponding Jacobian matrix. This, however, is not desirable, as the covariance matrices of each bearing vector are singular [53, 196]

	GPnP	UPnP	OGV non-linear
reference	[92]	[94]	[91]
observation	$\mathbf{v}$	$\mathbf{v}$	$\mathbf{v}$
linear complexity	yes	yes	no
multiple solutions	no	yes	no
central cameras	no	yes	yes
non-central cameras	yes	yes	yes
needs initial values	no	no	yes
points at infinity	no	no	no
$\mathbf{M}_t$ (MCS pose)	yes	yes	yes
$\mathbf{p}_i$ (reconstruction)	no	no	no
$\mathbf{M}_c$ (MCS calibration)	no	no	no
π_c^g camera self-calibration	no	no	no

(a)

BACS	MultiCol
[164] [163]	this work
$\mathbf{v}$	$\mathbf{m}'$
no	no
no	no
no	yes
yes	yes
yes	yes
yes	no
yes	yes
yes	yes
(yes)	yes
no	yes

(b)

Table 4.2. Compilation of algorithms for (a) only absolute pose estimation (b) structure from motion. Note that MultiCol can cope with both categories. However, MultiCol is not a minimal pose estimation method and requires initialization and linearization.

(projected from 2D observations) and thus the three rows are not linearly independent. Table 4.1 gives an overview of the two types of observations and the error functions different authors use.

One disadvantage of using image points as observations might be that the intrinsics have to be used in each iteration of the adjustment. This could be expensive to compute for larger sets of points. In [164] the authors use bearing vectors as observations to perform bundle adjustment. They reduce the number of observation equations back from three to two by using a reduced coordinate representation [53]. This requires the computation of the nullspace of each bearing vector and each scene point in each iteration. The corrections of the bearing vectors $\mathbf{v}$ are carried out in the reduced tangent space of each bearing vector (see [164] for details). The advantage of using spherically normalized bearing vectors and scene points is twofold. Using the homogeneous representation allows scene points to be infinitely far away from the MCS and the developed methods and algorithms are independent of the camera model, as only sets of bearing vectors have to be provided. Nevertheless, calibrated cameras are necessary in order to transform image plane observations to bearing vectors.

Our method uses image points and therefore relies on the intrinsics of the camera in each iteration. The advantage is that a simultaneous self-calibration as well as straightforward error modeling and propagation is easily possible. The covariance matrices of the observations are nonsingular as we optimize directly over the 2D image observations.

The non-linear optimization of [91] employs yet another residual function. They optimize Equation 4.21 by minimizing the dot-product between the measured and the reprojected bearing vector. Hence, they reduce the number of residuals for each measurement from two to one.

Table 4.2 gives an overview of all methods that are used in the simulations and experiments throughout the chapter. In Table 4.2a, variants of the NPnP that solve the non-central absolute pose) are listed. In contrast to GPnP [92], the improved UPnP [94] algorithm can handle central as well as the non-central cameras. Both algorithms are direct solvers and do not need initial values and are of O(n) complexity.

In addition, the library OpenGV ([91]) comes with a non-linear optimization method that minimizes the dot product between bearing vectors (dubbed OGV non-linear). All three absolute pose algorithms work on $\mathbf{v}$, i.e. bearing vectors instead of 2D image observations. None of those algorithms is tailored to perform MCS calibration, scene reconstruction or camera self-calibration. Hence π_c^g, $\mathbf{M}_c$ and $\mathbf{p}_i$ are not estimated. Nevertheless, we compare MultiCol against all three algorithms, as it is possible to solely refine $\mathbf{M}_t$ with MultiCol, i.e. the absolute pose of the MCS.

Table 4.2b depicts Bundle adjustment for multi-camera systems (BACS) and MultiCol. BACS performs bundle adjustment, hence $\mathbf{p}_i$ and $\mathbf{M}_t$ are estimated simultaneously. The public available implementation of BACS does not support the estimation of $\mathbf{M}_c$, but the authors show in [163] that this is of course possible as Equation 4.21 is used and can be linearized to include the camera to MCS transformation.

The self-calibration of all involved cameras is only possible with MultiCol. All other methods use the intrinsics of the cameras only once, to transform the 2D image measurements $\mathbf{m}'$ into bearing vectors $\mathbf{v}$.

4.4 MCS Calibration

The calibration of the MCS contains the definitions of the body frame and the subsequent estimations of all body to camera frame transformations $\mathbf{M}_c$ with $c = 1..C$, given image measurements. As recapitulated in Section 4.1, different approaches exist to calibrate a MCS with either overlapping or non-overlapping FoVs and static [108] or dynamic [55] constraints on the movement of the system. Our static, semi-automatic approach is applicable for non-overlapping MCSs and is also independent of two-view constraints assuming accurately synchronized cameras. In contrast to [108] it also inherently fixes the scale by calibrating from metrically known passpoints. We prepare an environment with circular black fiducial markers and measure their metric position (center) in an arbitrarily defined world frame using a motion capturing system. To extract the corresponding 2D image position, we perform region growing around a seed point within the black circle and subsequently fit an ellipse to the segmented image region. The ellipse center represents the 2D-measurement.

After recording $t = 1...T$ image tuples from different viewpoints, the following manual procedure is carried out for all tuples:

1. Initially select at least 4 points in *each* image c at time $t = 1$.
2. Estimate the exterior orientation $\mathbf{M}_{tc}^{-1}$ of each camera c at time $t = 1$ using an arbitrary central absolute pose algorithm [91].
3. Define the world to body frame transformation $\mathbf{M}_{t=1} = [\mathbf{R}, \mathbf{z}_0; \mathbf{0}^T, 1]_t$:
   (a) Initialize the rotation $\mathbf{R}_{t=1}$ to the inverse rotation of the first camera $\mathbf{R}_{11}^T$, i.e. $\mathbf{R}_{c=1} = \mathbf{I}$.
   (b) Set the translation vector to the mean translation vector extracted from all camera poses: $\mathbf{z}_{0t} = C^{-1} \sum_{c=1}^{C} \mathbf{z}_{0tc}$
4. Extract all body to camera frame transformations $\mathbf{M}_c = \mathbf{M}_t^{-1} \mathbf{M}_{tc}^{-1}$

This procedure basically separates the exterior orientation $\mathbf{M}_{tc}^{-1}$ of each camera into two transformations, i.e. the world to body and the body to camera transformation. The last step of the initialization procedure yields approximate values for the body to camera frame transformations ($\mathbf{M}_c$).

Finally, MultiCol is applied to simultaneously refine all MCS poses $t = 1..T$ and $c = 1..C$ body to camera frame transformations. For the following simulations and evaluations, all $\mathbf{M}_{c=1..c}$ are assumed to be already estimated and fix.

4.5 Experiments and Results

To evaluate and test the performance of our method, we perform several simulations. First, MultiCol is evaluated against absolute pose algorithms, i.e. the GPnP (linear, non-iterative NPnP), UPnP (universal PnP and NPnP) and the non-linear estimation algorithm that optimizes over bearing vectors. All algorithms are implemented in OpenGV [91] and are available online. Note that MultiCol is not a direct O(n) solution to the NPnP problem, but a different non-linear refinement method. Afterwards, we simulate different bundle adjustment configurations and compare MultiCol against BACS [164].

All simulation parameters were chosen to mirror realistic conditions that arise in the context of this work, e.g. the extent of the scene or the number of points. The MCS poses are sampled within the volume defined by the extent of the scene points.

4.5.1 Simulations

Pose estimation

We compare four MCS configurations that were explored in [92]. In addition a fifth, completely random setup is tested, i.e. the MCS configuration is re-sampled in each iteration. We further add a sixth experiment. Far points are added to the scene, i.e. points that are more than $\pm 500m$ and up to $\pm 5000m$ away from the MCS. Overall, the following MCS configurations are explored:

1. a front and a back looking camera
2. a stereo configuration
3. one front and one side looking camera
4. one front, one back and two side-looking camera
5. three cameras, random configuration
6. three cameras, random configuration and far object points

For the random configuration, the camera offsets from the body frame are sampled up to $0.5m$.

To assess the algorithms we compute the median orientation and translation error. Latter is calculated as the euclidean distance between known and estimated translation vector. To get the rotational error in radians we transform the known and estimated rotation matrices to their corresponding angle-axis Rodrigues representations (Section 2.2.1.4). Then, we can compute the norm of the angular difference in this representation.

The pose estimation is iterated $n = 1000$ times for each noise level. The scene consists of $I = 50$ scene points $\mathbf{p}_i$ that are sampled in a volume of $50m \times 50m \times 50m$.

Figure 4.3 depicts the median translation and orientation errors for the four static configurations (1.-4.). In each case, MultiCol yields a higher accuracy than GPnP, UPnP and the corresponding non-linear optimization proposed in [92], that minimizes the dot product between bearing vectors (see Table 4.1). The same holds for the random configuration case Figure 4.4. Adding far points to the scene, does not significantly change the accuracy of the non-linear and the MultiCol case, but significantly decreases the accuracy of the GPnP as well as the UPnP algorithm.

To investigate this behavior, we conduct a seventh experiment. We now fix the image noise to 1 pixel and increase the volume in which scene points are sampled. Figure 4.6 depicts the results. The pose estimation of the two O(n) algorithms, i.e. GPnP and UPnP is sensitive to the extent of the scene and the distance of the scene points from the camera system. In contrast, the orientation accuracy, remains constant. The UPnP algorithm performs a lot better. It seems that adding only a couple of far points to a small scene affects the algorithms more, than a homogeneous enlargement of the whole scene.

In Section 4.3, the differences between MultiCol and the methods of [91] were elaborated. Recall, that [91] minimizes over the dot product of reprojected bearing vectors, whereas MultiCol minimizes the reprojection error in the image plane. The latter appears to be computationally more intensive, because the points have to be projected to the image plane in each iteration. Figure 4.6 reveals, that this is not the case for realistic sized problems. Especially the convergence behavior (Figure 4.6e) of MultiCol is superior to the optimization over dot products. The small number of iterations fully compensates for the additional effort of projecting the points to the image plane.

Structure-from-Motion

The second simulation refers to a structure from motion (SfM) approach. In contrast to the mere pose estimation performed in the last experiment, we now perform simultaneous scene reconstruction as well and the results are compared to our reimplementation (in C++) of the online available BACS [164]. The correctness of the reimplementation was verified with the provided Matlab code of the authors.

In this experiment, we tried to mirror a SLAM like local bundle adjustment problem, as this will be of interest in Chapter 7. Again, we sample a scene now consisting of $I = 200$ scene points $\mathbf{p}_i$ in

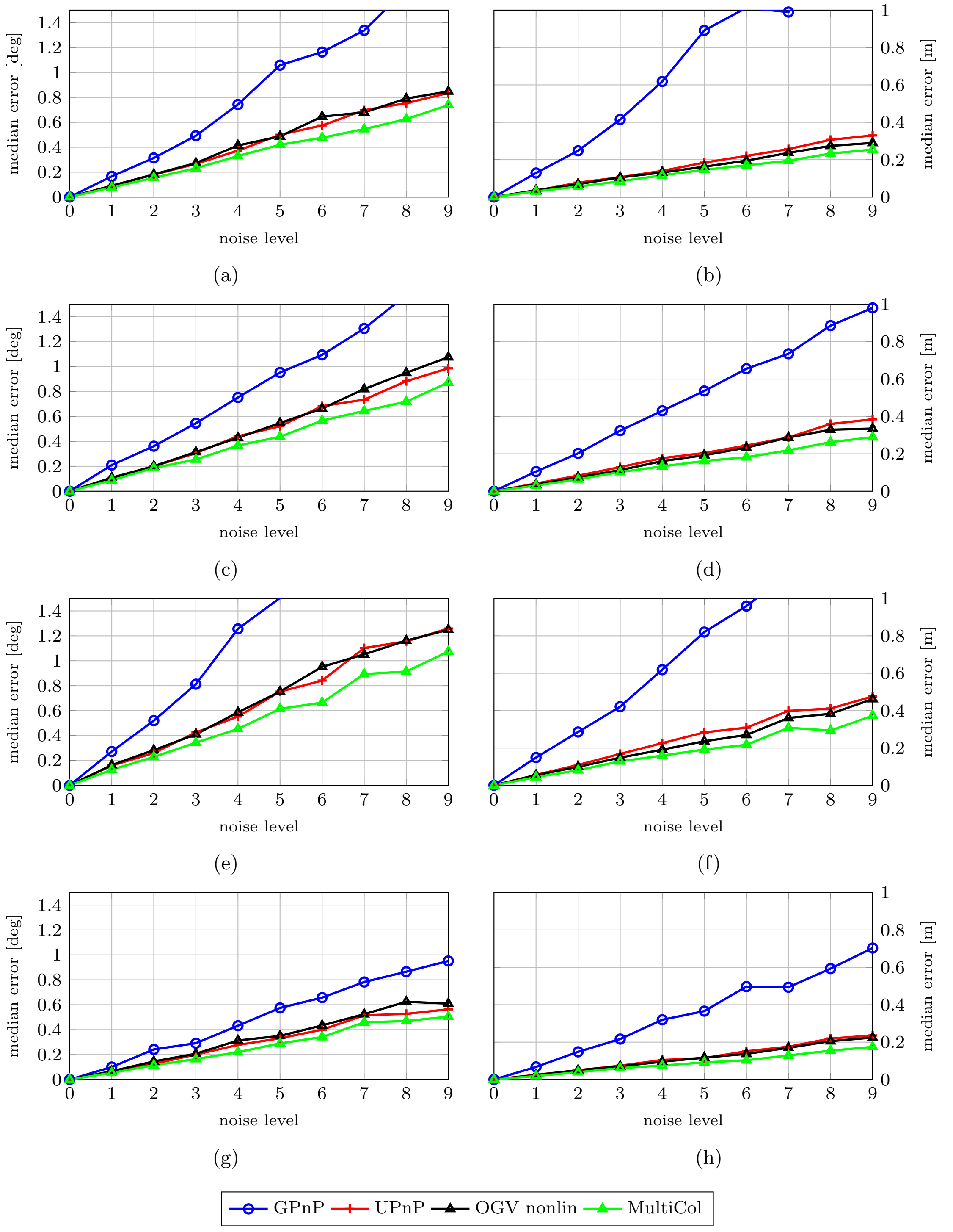


Figure 4.3. Median rotation error (column 1) and median translation error (column 2) for the pose estimation and different MCS configurations. (a)(b) configuration 1, one front and one back looking camera (c)(d) configuration 2, stereo configuration (e)(f) configuration 3, one front and one side-looking camera, (g)(h) configuration 4, one front, one back and two side-looking cameras

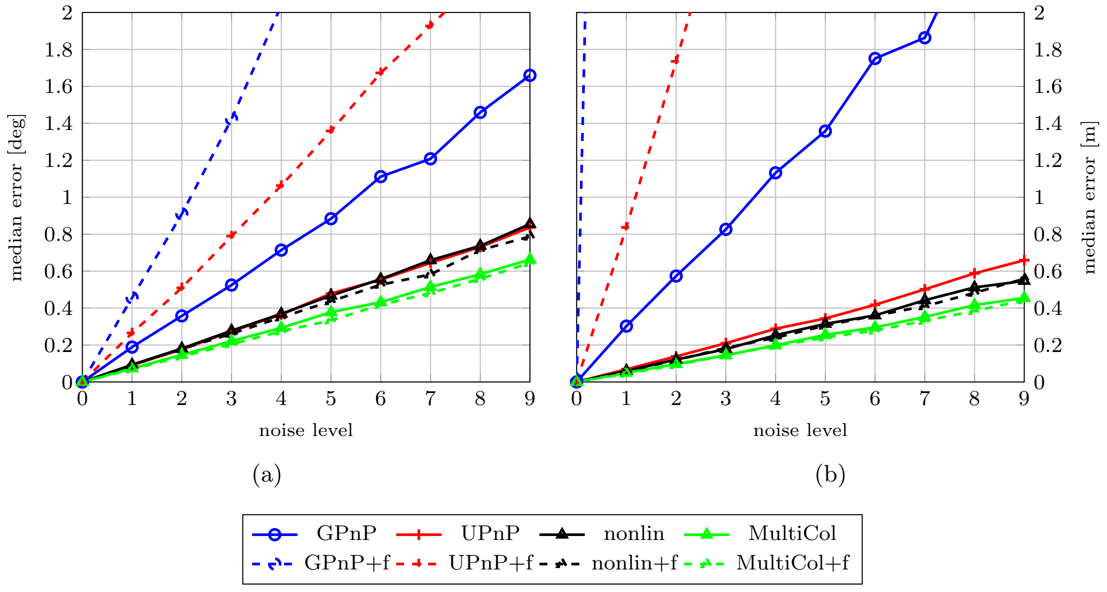


Figure 4.4. Median rotation and median translation error of MCS pose estimation respectively for the random configuration and far scene points case (+f). The latter is depicted in dashed lines.

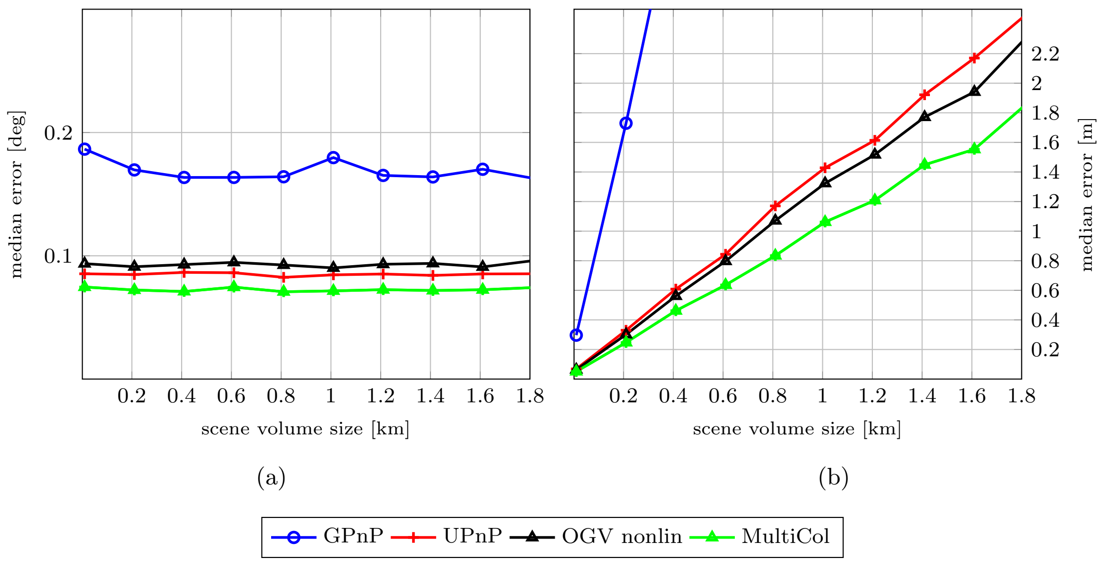
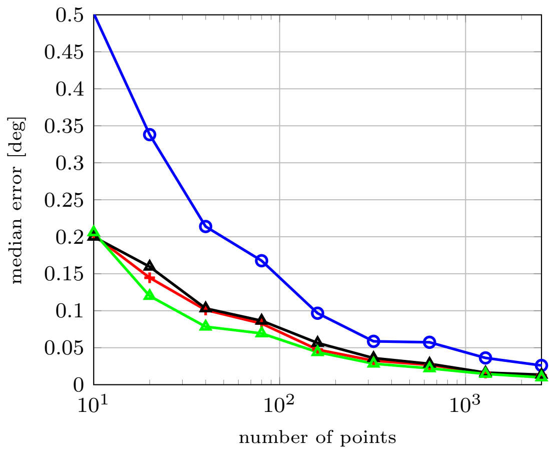
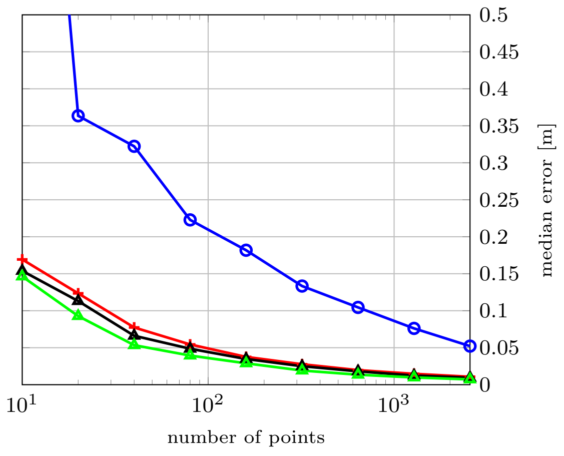


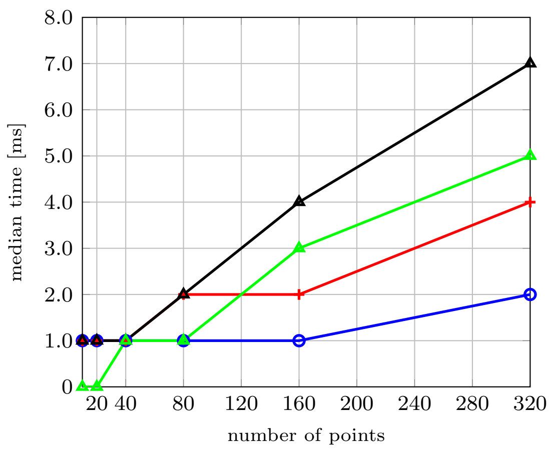
Figure 4.5. Median rotation and median translation error of MCS pose estimation respectively. The noise level is fixed to one. This time the volume in which scene points are sampled is increased.



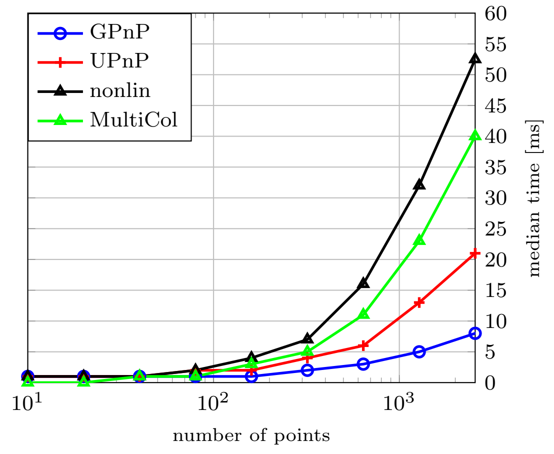
(a)



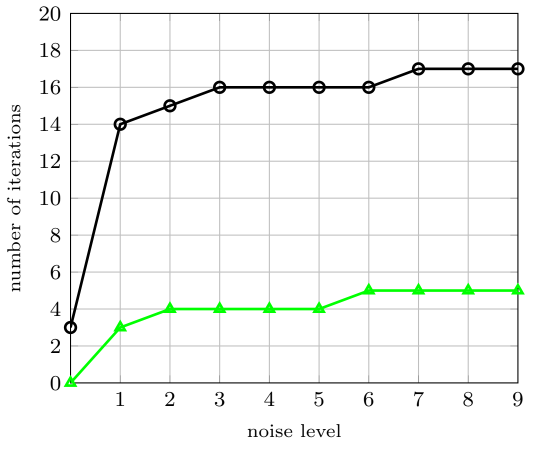
(b)



(c)



(d)



(e)

Figure 4.6. Median rotation and median translation error of MCS pose estimation respectively. Here, the point number is steadily increased. In addition, we examine the convergence behaviour and the total computation time. For (a) - (e) the noise level is fixed to one. (a) and (b): median rotation and median translation error respectively. (c) median runtime for up to 320 points. (d) median runtime for up to 2500 points. (e) median number of iterations until convergence.

a volume of $50m \times 50m \times 50m$. The MCS configuration ($\mathbf{M}_c$) is always randomly sampled in each iteration. For each simulated scene $T = 20$ MCS poses $\mathbf{M}_t$ are randomly distributed in the scene volume. In addition, a simulation that contains far points is conducted.

Figure 4.7 depicts the median rotation (a), translation (b) and reconstruction error (c). Note, that each median error is calculated twice, i.e. the first is the median error over the all MCS poses. The second median error is calculated over all iterations ($n = 200$) of one particular noise level. The translation error increases linearly with the noise level for all four cases, i.e. BACS and MultiCol and both with 10% ratio of far points (we added 20 far points). The same holds for the rotation error.

The reconstruction error computes as follows. We compute the mean Euclidean distance between all ground truth and reconstructed scene points. To eliminate degenerate configurations, we then calculate the median of the mean Euclidean distances over all iterations.

4.5.2 Interior Self-Calibration

The last simulation evaluates the simultaneous self-calibration of all involved cameras. This means that we augment the Jacobian with a parameter block for the interior orientation of each camera. The full parameter block for the intrinsics consists of the three affine parameters, the two components of the principal point, as well as the polynomial coefficients of $\rho(\theta)$. Instead of calibrating the forward projection function $f(\rho)$ (see Chapter 3.3) here, the coefficients of the back-projection polynomial are refined. As it is usually difficult to calibrate parameters that do not have a physical meaning (like the coefficients of $\rho(\theta)$), we split our investigation into two parts, i.e. we estimate the intrinsics with and without the polynomial.

In total, we conduct four tests:

1. MultiCol without intrinsics
2. BACS without intrinsics
3. MultiCol with principal point and affine parameters
4. MultiCol with full intrinsics estimation

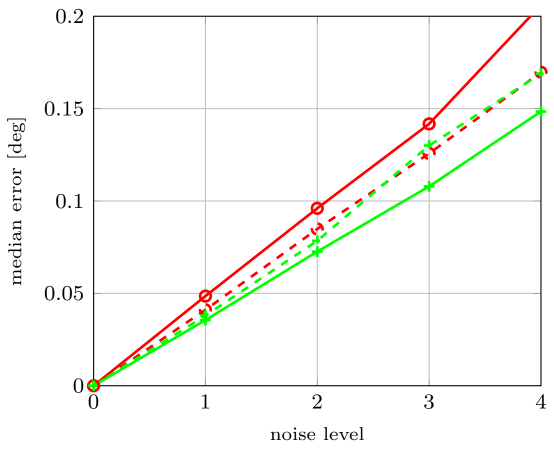
This time, we sample $T = 20$ MCS poses as well as the $I = 100$ scene points in a $20m \times 20m \times 20m$ volume and repeat the simulation $n = 50$ times for each noise level. We increased the number of poses, to get a higher redundancy and thus better conditioned normal equations. Further, we decreased the size of the volume to increase the probability of having image measurements distributed over the whole image of each camera.

Bundle adjustment is carried out as a free network adjustment. Thus, possible tensions between the scene points will not affect the calibration of the intrinsics.

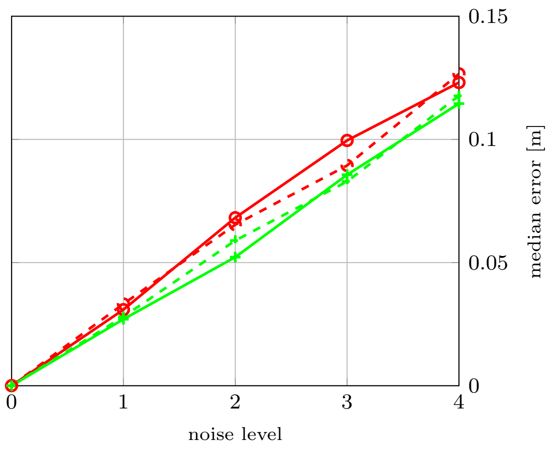
To simulate varying intrinsics, we add Gaussian noise to the principal point as well as the affine parameters. Then the forward projection function $f(\rho)$ is perturbed (scaled by the magnitude of the coefficients) and the inverse polynomial $\rho(\theta)$ is calculated.

The results for the test cases are depicted in Figure 4.8. The top row shows the median errors for orientation and translation of the MCS pose. As expected, the performance decreases if the interior orientation is not considered. Then, the principal point as well as the affine parameters are added to the unknowns, simulating a self-calibration of the cameras, e.g. after small lens movements. Even if the polynomial coefficients are omitted, this step yields a significant enhancement. In a last step, the parameter block for the inverse projection function is added, again, yielding a performance increase.

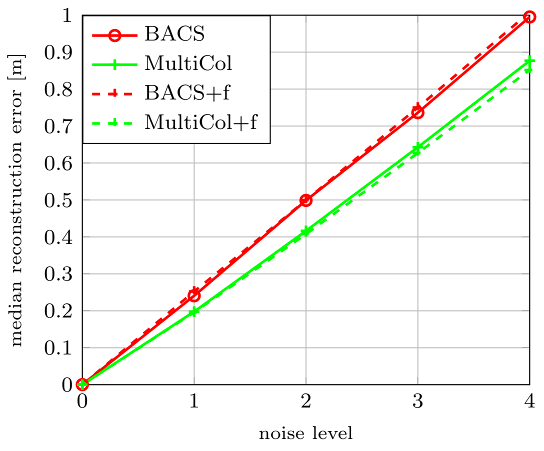
Figure 4.8c depicts the distance of the re-estimated principal point to the ground truth value. The largest difference is, of course, MultiCol without intrinsics. As expected, self-calibration including the polynomial yields the best results. Hence, the calibration of the inverse projection function is, to some extent, possible. However, if the bundle has only small redundancy, the image observations are poorly distributed over the image, or the scene configuration is degenerate, the normal equations will be ill-conditioned and the inversion unstable. Figure 4.8d depicts a case, where the distribution



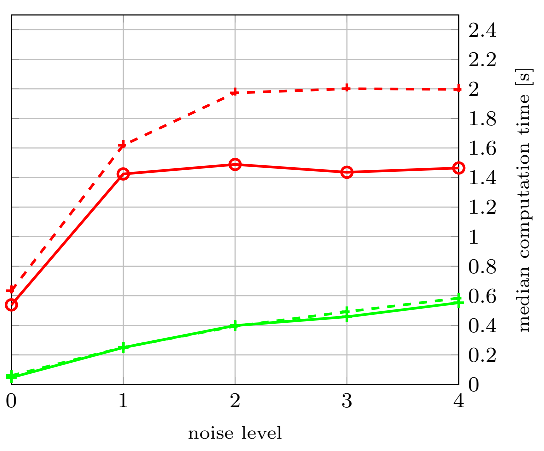
(a)



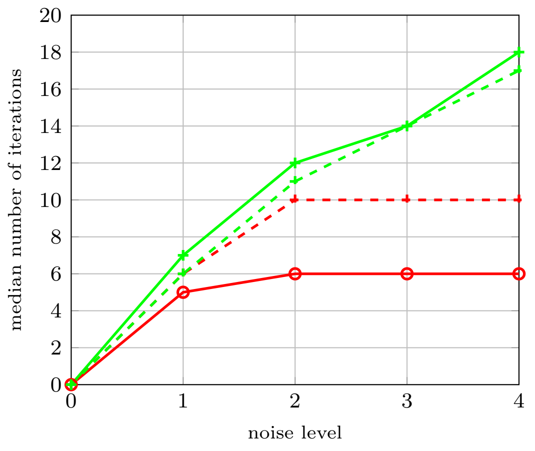
(b)



(c)



(d)



(e)

Figure 4.7. Comparison of bundle adjustment simulations between BACS and MultiCol. Each far point (+f) is sampled randomly on the interval of 1000m to 5000m. (a) - (b) Median orientation and median translation error respectively. (c) Median reconstruction error. Far points are omitted in the calculation. (d) Median computation time for the bundle adjustment. (e) Corresponding median number of iterations.

of points along the mirror borders was marginal. A point with incident angles of $-\pi/2$ would have been observed along the principal axis. A point perpendicular to the principal axis has an incident angle of 0. The re-calibrated polynomial starts to be unstable for points close to the image border, and the exact re-calibration of the original polynomial in this image area is hardly possible.

4.5.3 Real Data

After evaluating the MCS algorithms in simulated scenes, we conduct experiments in a real environment, to assess the performance on real data. The goal of our evaluation is to assess not only the performance of the pose refinement, but also of the scene reconstruction. Hence, accurate ground truth for MCS poses as well as the scene are necessary. Numerous data sets for the evaluation of SLAM algorithms exist, covering large areas and consisting of stereo or single cameras e.g. [17, 63, 165]. Indoor datasets with accurate ground truth are available from [29, 180] to name a few, but again not containing any MCS pose and scene ground truth. As we are not aware of an accurate ground truth dataset for non-overlapping MCSs consisting of poses, calibration data and scene points, we recorded six datasets with different motion characteristics in an indoor environment.

1. fast, steady forward motion and slower, wavy backward motion ($T = 1019$)
2. fast, steady foward motion along the whole corridor ($T = 680$)
3. rotation on the spot about the z-axis ($T = 200$)
4. no motion ($T = 650$)
5. circular motion in the x-y plane ($T = 440$)
6. up and down motion along the z-axis ($T = 250$)

The first two were already described in Section 2.5.

For each data set the trajectory $\mathbf{M}_t, t = 1..T$ of the MCS is given in two ways. The first is the ground truth trajectory $\mathbf{M}_t^{gt}$. The corresponding MCS trajectory $\mathbf{M}_t^{MCS}$ is estimated by tracking passpoints (black circles) over time.

The 3D position of each passpoints was measured with the T-Probe in the system of the lasertracker with an uncertainty of about $\sigma_{pos} = 0.1$mm. From an initial pose, we project each passpoint into all camera images using Equation 4.18. Then a search for a homogeneous dark region around that potential 2D location is performed. If such a region is found, the minimal bounding ellipse is calculated and the center of that ellipse is used as the corresponding 2D observation $\mathbf{m}'_{itc}$. In this way, we create a set of potential 2D image observations for each MCS pose.

This simple brute-force search yields outlier image measurements, e.g. when two projected passpoints are very close together or behind a dark object. To cope with outliers in the data, basically two methods exist. The first is to use iterative re-weighted least squares (IRLS), where the influence of outliers is down-weighted in each iteration. MultiCol is implemented as IRLS using Huber's weight function. This method is abbreviated with '+ m' in Figure 4.9. The second are iterative random sampling methods such as RANSAC) where a minimal set of the data is used to fit a model in each iteration. In our experiments, we use RANSAC in combination with GP3P [92], which is part of OpenGV. This method is abbreviated with '+ r' in Figure 4.9.

We compare MultiCol against two absolute pose algorithms, namely UPnP which is a direct O(n) solution that needs no initialization and the non-linear refinement of OpenGV. The latter optimizes over the angle between bearing vectors whereas MultiCol optimizes the reprojection error (cf. Table 4.1). For all experiments (except MultiCol + m) the outliers are removed a-priori using RANSAC + GP3P. Then the resulting pose of GP3P is used to initialize the subsequent refinement (MultiCol and the non-linear refinement of OpenGV). For MultiCol + m the previous trajectory pose $\mathbf{M}_{t-1}^{MCS}$ is used to initialize the refinement.

To compare all algorithms, we calculate the relative orientation between the MCS pose and the T-Probe pose which should be constant over time, i.e. for each pose $t = 1..T$. The transformation between both sensors is: $\mathbf{M}_t^{rel} = (\mathbf{M}_t^{gt})^{-1}\mathbf{M}_t^{MCS}$. From $\mathbf{M}_t^{rel}$ the rotation (3 parameters) as

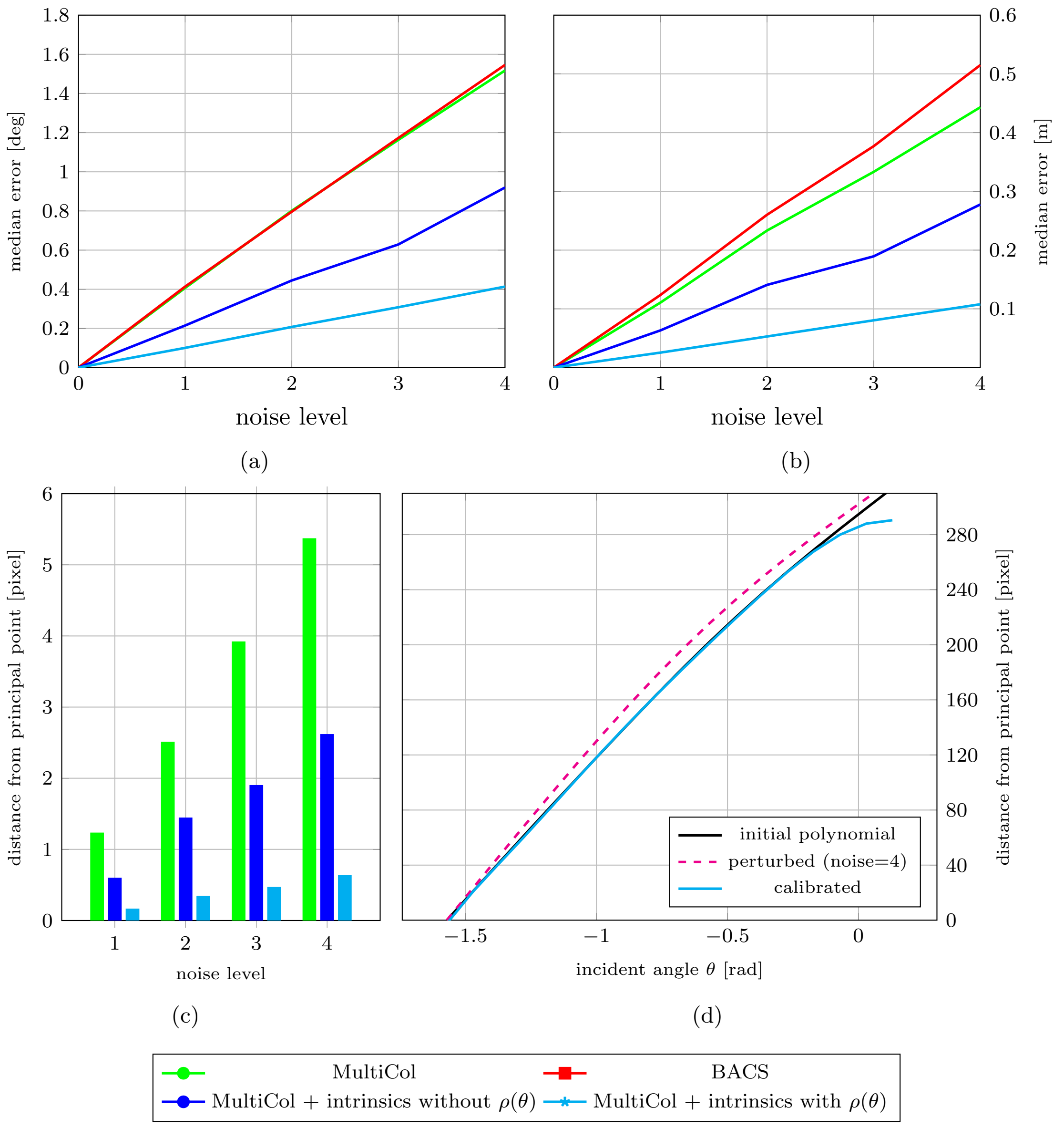


Figure 4.8. Performance evaluation of simultaneous self-calibration. Noise is added to the scene points, the MCS pose and now in addition to the intrinsics of each camera. (a) and (b) depict the median orientation and translation error of the MCS pose with increasing noise. (c) depicts the distance to the ground truth principal point after self-calibration. (d) depicts the original, perturbed and re-calibrated polynomial for all possible incident angles. Incident angle of $-\pi/2$ is exactly along the principal axis. The cameras have a field of view of about 185°, i.e. the maximum incident angle is ≈ 0.05 rad.

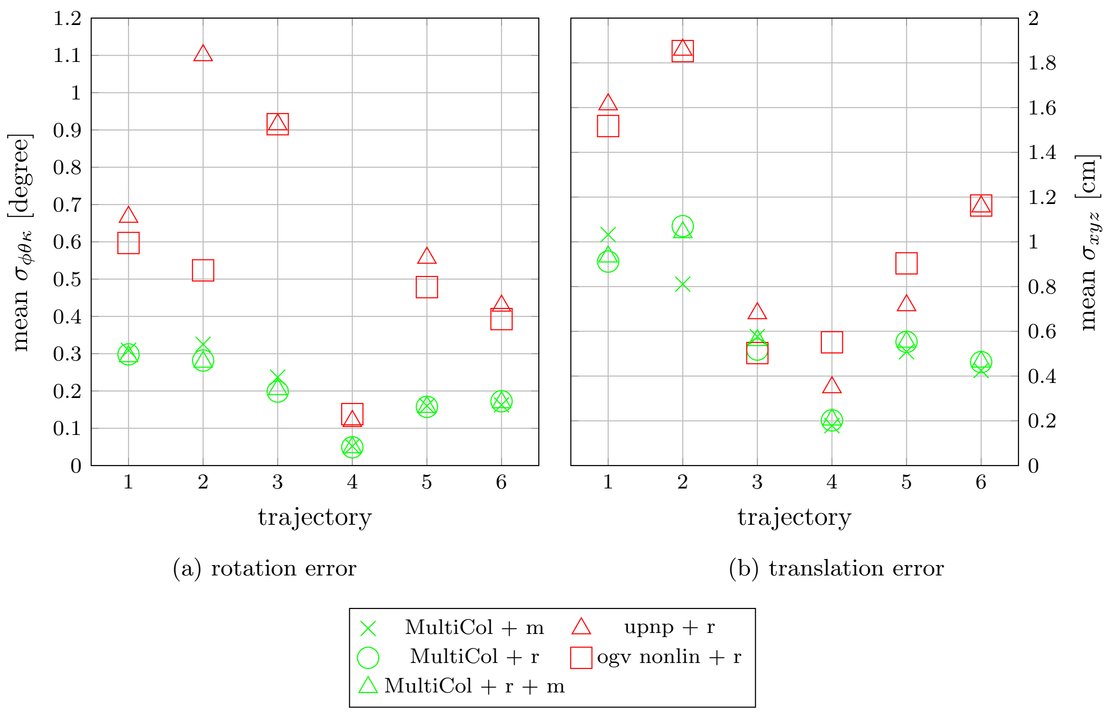


Figure 4.9. Comparison of all six trajectories. (a) depicted is the mean standard deviation of the three rotation angles. (b) depicted is the mean standard deviation of the three translation elements.

well as the translation (3 parameters) can be extracted for each t. Having estimated the relative orientation T times, we can subsequently calculate the standard deviation for each parameter, i.e. σ_x, σ_y, σ_z, σ_ϕ, σ_θ, σ_κ. Figure 4.9a depicts the mean standard deviation for the 3 orientation elements and Figure 4.9b the mean standard deviation for the 3 translation elements.

The experiments confirm the simulation results. MultiCol performs better for each trajectory. It also shows, that the outlier removal using RANSAC is not absolutely necessary if IRLS is used. MultiCol + m performs as well as MultiCol + r.

Another problem that came up with UPnP and nonlin which is not depicted in Figure 4.9 is, that for some poses both algorithms yielded completely wrong poses (probably due to outliers that remained in the data despite RANSAC). For those cases, where the deviation of the relative orientation was above some threshold, we set the relative pose to the MultiCol result. Otherwise the results would have been significantly worse.

To evaluate the SfM performance of our method, we use the ground truth trajectories 1 and 2 and manually select poses and corresponding image measurements for each pose. Table 4.3 summarizes the amount of points and poses that were used to evaluate MultiCol against BACS. For both trajectories, all points were used to define the datum, as the extent of the scene is rather small. To reduce the influence of outliers, we again perform IRLS.

To simulate noisy approximate values, we add Gaussian noise to both poses and scene points. The scene points are perturbed with a standard deviation of 5 cm. The pose is perturbed with a standard deviation of 1cm and 0.1 rad for translation and orientation respectively.

Figure 4.10 depicts the results for trajectory 1 and 2. Both datasets were evaluted with MultiCol and BACS respectively. It shows that the results confirm the structure from motion simulations. Both methods perform similar. The first two rows of Figure 4.10 depict the results for the motion part. For the translation part of the pose, MultiCol seems to be slightly inferior. The simulation results (see Figure 4.7) gave the impression, that MultiCol yielded better results for the orientation of the MCS. This fact can not be verified with the ground truth data set. Both algorithms yield almost equal angular accuracy (see Figure 4.10b and Figure 4.10a).

	trajectory 1	trajectory 2
# poses ($\mathbf{M}_t$)	27	36
# scene points (i)	30	47
# cameras (c)	3	3
# observations $\mathbf{m}_{itc}$	475	504
redundancy	705	658

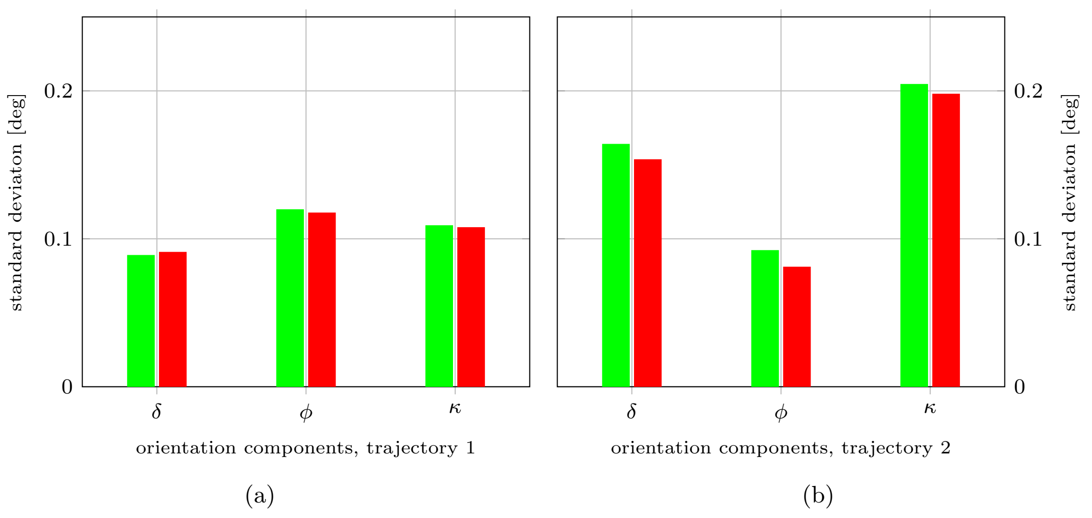
Table 4.3. Selected data from laser trajectory 1 and 2 (Section 2.5) that is used to evaluate the SfM performance of MultiCol and BACS.

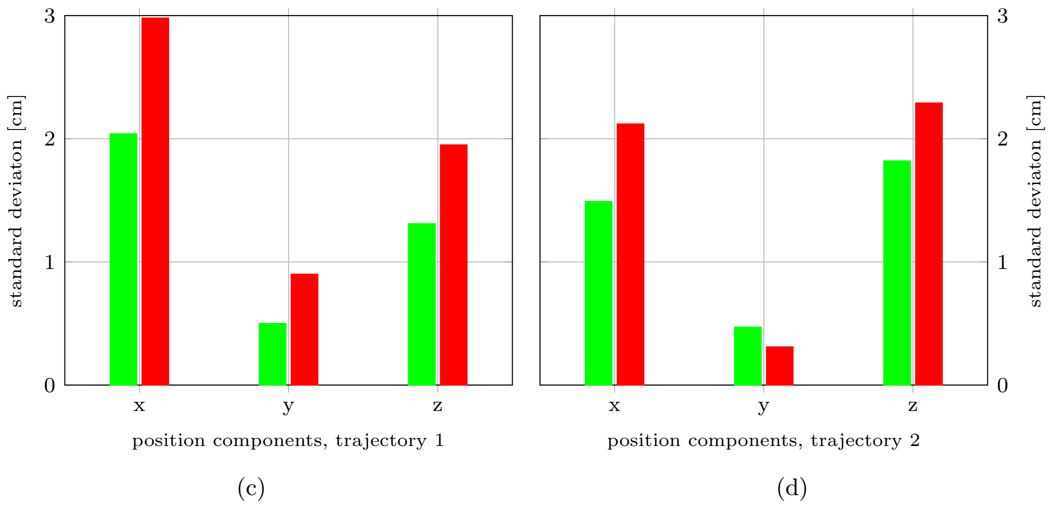
As MultiCol and BACS reconstruct the structure of the scene as well, we investigate the reconstruction error of the scene points. We calculate it as the distance of the reconstructed point to the respective unperturbed passpoint. The last row of Figure 4.10 depicts a histogram of the reconstruction error for each trajectory. Here, MultiCol seems to yield better results as BACS, because more points are cumulated in the smaller error bins.

4.6 Summary

In this chapter, we presented a general and modular method for pose estimation, simultaneous self-calibration and scene reconstruction for multi-camera systems called MultiCol which is based on the extension of the classical collinearity equations. First, we outlined differences to other methods concerning types of observations and error modeling. Then, we performed numerous simulations and evaluated MultiCol against publicly available methods. For mere pose estimation, MultiCol was evaluated against the absolute pose methods GPnP, UPnP and the nonlinear refinement over the dot products of bearing vectors that are all part of OpenGV ([91]). To evaluate the SfM performance, we compared our approach to BACS ([164]).

Our method shows similar performance in all cases and is slightly superior in most cases. The latter does not only hold for the accuracy, but also the convergence rate and thus the runtime. The modular implementation and the inclusion of a general camera model makes it highly modular and flexible. It can either be used to calibrate the extrinsics of the camera system, the instrinsics of all cameras or it can be used to estimate the pose of the system at the same time.





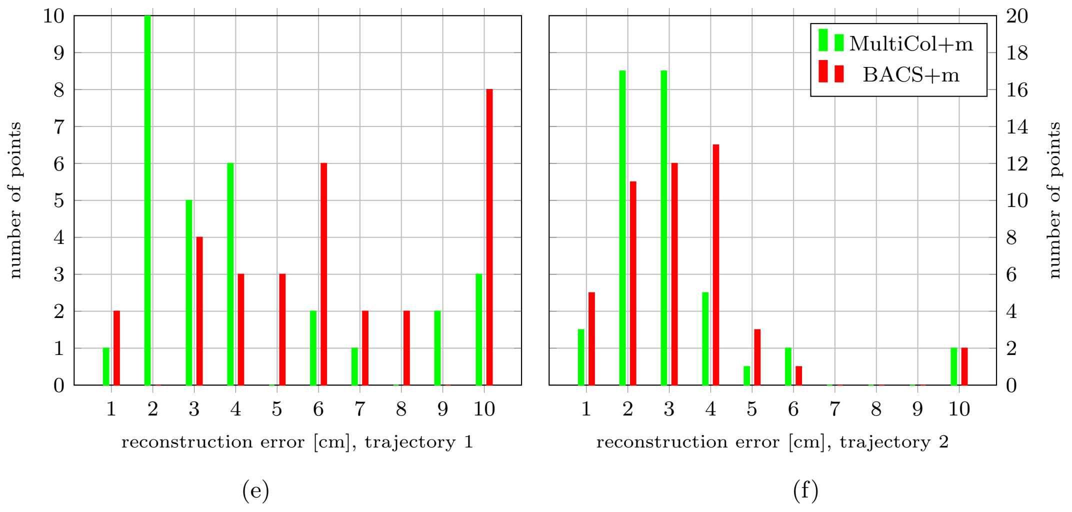


Figure 4.10. Depicted is the result of the ground truth evaluation for trajectory 1 (left column) and trajectory 2 (right column). (a) and (b) depict the standard deviation of the 3 orientation elements. (c) and (d) depict the standard deviation of the 3 translation components. (e) and (f) depict a histogram of reconstruction errors, i.e. how much points have a reconstruction error less than the bin value. Scene points with an error greater than 0.1m are omitted.

5. Online Adapted Binary Descriptors for Omnidirectional Images

The last two chapters have set the geometry related foundations for multi-camera motion and structure estimation. In this chapter, the focus lies on the robust detection and description of salient point features. This step is at the core of most computer vision or photogrammetric processing chains, e.g. object detection, image retrieval, 3D reconstruction, localization, scan registration [198], and numerous methods have been developed over the past decades. Most features can be grouped into different categories (see Table 5.1 and Table 5.2) and all have their advantages and disadvantages, depending on the task, the application or the platform. Commonly, the feature extraction stage can be divided into three main steps.

Detection

First the image is searched pixelwise for keypoints. Depending on the *detector*, those keypoints can either lie on corners or blob-/region-like structures. A desirable property of such detectors is that they should be able to repeatably detect the same keypoint over viewpoint changes. In addition, a robust estimation of the orientation as well as the scale at which the keypoint was detected is necessary for some applications.

Description

After having identified keypoints, a robust description of the image content surrounding a keypoint is performed. The description of an image patch is usually mapped into a vector which is called a *descriptor*. Depending on the method, such descriptors consist either of floating point values or binary strings. If the keypoint detector provides rotation or scale information, the descriptor can be made invariant to such viewpoint changes. For many tasks including ego-motion estimation this property is of fundamental necessity as the movement of the camera introduces a continuous change of viewpoint.

Matching

In the last step, descriptors are matched by comparing their vector representations. A match is found if the distance between two descriptors is minimal compared to all other descriptors. Depending on the vector type either the L2-Norm (floating point) or the Hamming distance (binary strings) can be computed. Latter has the advantage of speed, as the instruction sets (SSE, AVX) of modern processors allow for parallel bit counting. In addition, different matching strategies exist. One can either match each descriptor from one image to each descriptor from another image, or apply fast approximate nearest neighbor algorithms [134].

At this point it is important to emphasize that there does not exist one perfect feature for all tasks. Some descriptor might outperform all others in terms of recognition rate on one dataset, but might

Method	Detector	Blob/Corner	Scale/Orientation/Distortion	Time μs/Point
AGAST	AGAST	corner	-/-/-	0.48
FAST	FAST	corner	-/-/-	0.84
ORB	FAST on pyramid	corner	✓/✓/-	19
BRISK	AGAST on pyramid	corner	✓/✓/-	24
SPHORB	SFAST	corner	✓/✓/✓	> ORB
SURF	SURF	blob	✓/✓/-	47
A-KAZE	A-KAZE	blob	✓/✓/-	152
SIFT	DoG	blob	✓/✓/-	170
sRD-SIFT	distorted DoG	blob	✓/✓/✓	> SIFT

Table 5.1. Overview of state-of-the-art keypoints detectors. Except for sRD-SIFT and SFAST, all algorithms are part of OpenCV 3.1.

be impractical for real-time applications due to a high computational burden. Also this chapter will not develop a new point feature for omnidirectional images from scratch. In the first section we will recall the state-of-the-art in image feature research and group it into different categories that are of interest in the context of this dissertation. The next two sections will then extend and adopt recent approaches for the use in online omnidirectional image matching. Finally the last two sections will give results on matching datasets and a conclusion.

5.1 State-of-the-Art

This section gives an overview of the state-of-the-art for feature point detectors and descriptors. As this is a vast topic, we can only cover a small amount of related work. Instead, this section is supposed to analyze the currently available methods and help to find, adapt and tune a good feature detector - descriptor combination to the given task of multi-fisheye camera motion estimation. For more comprehensive overviews and evaluations of recent research and early developments, the reader is referred to [49, 62, 74, 127, 128, 203].

To narrow down the state-of-the-art relevant to the task addressed in this work, lets first analyze the requirements that our detector-descriptor combination has to meet. First of all, the 3-stage process of extraction, description and matching has to be performed in real-time. For a multi-camera system running at 25 Hz, we have at most $40ms$ for each image tuple to be processed. From those $40ms$ at least $10ms$ have to be subtracted for pose estimation (Figure 4.6c) and other processing steps. If there is more than one camera, the remaining $30ms$ had to be divided by the number of cameras, although there is some room for parallelization. In addition, the descriptors are supposed to be used in place recognition and relocalization tasks and thus the storage requirements are not negligible as a database of descriptors has to be build. Apart from the processing speed, the detector-descriptor combination should be invariant (to some extend) to viewpoint and illumination changes, as the motion of the camera system is unconstrained. Latter however allows to relax the requirements on the amount of robustness to significant viewpoint change, as the motion (and thus baseline and rotation) between subsequent frames is rather small and the putative location of the features can be reliably predicted from the last camera poses. Moreover, the descriptors should be invariant under image distortion or fisheye projections. As we will see, latter is a requirement rarely addressed. As the camera system is supposed to operate in a changing environment and to visit the same scene more than once, an online adaption of the descriptors that are stored in the database would also be favorable.

5.1.1 Keypoint Detectors

In general, keypoint detectors can be grouped into two categories, i.e. corner and blob-like detectors.

Corners

Two of the first corner detectors, still in use in many applications, are the Harris corner [72] and Shi-Tomasi's "Good Features to Track" [169] detector. In both methods, a "corner score" is derived by analysis of the eigenvalues of the second-order moment matrix which involves computing the image derivatives. To avoid such costly computations Features from Accelerated Segment Test (FAST) [152, 153] uses simple gray value comparisons on discretized circle segments around a candidate pixel to decide whether a pixel is classified as a corner. To speed up the process, they involve machine learning techniques (a ternary Decision Tree) to reduce the number of necessary pixel comparisons to a minimum. The drawback is that this Decision Tree should be relearned for new environments or scene structures. Building upon the idea of FAST, the Adaptive and Generic corner detection based on the Accelerated Segment Test (AGAST) [119] detector improves the segment testing by finding an optimal binary Decision Tree that adapts to new environments. In addition, the ternary test configuration of FAST (similar, brighter, darker) is extended to be more generic. So far, none of the detectors is invariant to scale changes or estimates a salient orientation. The Oriented FAST and Rotated BRIEF (ORB)[154] detector builds an image pyramid and extracts FAST features on every level. To avoid large responses along edges, the authors use the Harris corner score on every keypoint. In addition, the magnitude of the corner measure is used to order the keypoints. However, no non-maxima suppression is performed between different scale levels, leading to potential duplicate detections. To estimate a salient orientation, an intensity centroid is computed in the patch. Then, the orientation is defined by the vector between the patch center and this centroid. The Binary Robust Invariant Scalable Keypoints (BRISK) [106] detector tries to improve upon ORB by extracting AGAST corners on different levels of the image pyramid and scale interpolation between octaves. In addition, a non-maxima suppression across scales is performed, using the FAST score as a measure of saliency. The orientation is estimated from the gradients of long-distance pairs of a distinct sampling pattern around the keypoint that is later also used to calculate the descriptor.

Blobs

Blobs are usually detected as the local extrema of different filter responses. The Scale Invariant Feature Transform (SIFT) [115] detector generates a scale space by subsequent filtering of the image with separable Gaussian filters with increasing variance. Then, extrema are detected in Differences of these Gaussian filtered images. To speed up the computation, the Speeded Up Robust Features (SURF) detector [16] approximates the Gaussian derivatives by box filters, thus circumventing the creation of the whole Gaussian scale space by simple upscaling of the filter size. In addition, both detectors estimate a patch orientation. SIFT builds a histogram of gradient orientations to select the one that is dominant in the patch. SURF uses the sums of Gaussian weighted Haar Wavelet responses in two directions to assign an orientation to the keypoint. The authors of the KAZE [5] detector as well as its accelerated pendant A-KAZE [6] argue that a linear Gaussian scale space smooths noise and object details to the same degree. Hence, they build a non-linear scale space using non-linear diffusion filtering to obtain keypoints that exhibit a much higher repeatability. The keypoint orientation is found using a method similar to the SURF detector.

So far, none of the presented detectors considers image distortion or different projections. Both introduce deformations to corners and blobs that will decrease the repeatability of the detector. In theory, the image could be warped to match a perfect planar perspective projection. This however introduces severe artifacts and is limited to images with less than a hemispherical view of the scene and should be avoided [37]. Thus, all image processing operations have to be carried out directly in the distorted image. To detect repeatable keypoints in omnidirectional images, invariant to scale and rotation, the construction of the scale space [22, 68, 146] as well as the image gradient computation have to be adjusted [57]. Prominent enhancements of the SIFT blob detector are pSIFT [68] and sRD-SIFT [114]. They excel their "standard" pendants regarding matching performance but the adapted construction of scale-spaces on the sphere increases their extraction time significantly and makes them even less suitable for real-time applications.

The same issues occur with corner detectors. FAST, for example, takes N pixels of a geometric shape (a Bresenham circle) around each candidate keypoint and compares their intensities. This works well, assuming that the pixel neighborhood remains the same over the image domain. Certainly, if the image is affected by strong distortion, the pixel neighborhood varies depending on position from the image center. Thus, the circle had to be adjusted (distorted) depending on the position of the point in the image and pixel intensities would have to be interpolated. This, however, destroys the high efficiency of the detector. For SPHORB (Spherical ORB) [217], the authors introduce the geodesic grid, borrowing ideas from climate modeling [147]. Basically, the sphere is subdivided into Voronoi cells of similar size. Now, each pixel can be assigned to Voronoi cells

Descriptor	type	scale/rotation/distortion	#byte	speed μs/point	off-line learned	online adaption
BRIEF	(IV) binary	✓/✓/-	16-64	6	-	-
ORB	(IV) binary	✓/✓/-	32	14	✓	-
BRISK	(IV) binary	✓/✓/-	64	20	-	-
BOLD	(IV) binary	✓/✓/-	64	88	✓	✓
BinBoost	(IV) binary	✓/✓/-	8	420	✓	-
TailoredBRIEF	(IV) binary	✓/✓/-	16-64	-	-	✓
SPHORB	(IV) binary	✓/✓/✓	32	-	✓	-
SURF	(II) real-valued	✓/✓/-	256	76	-	-
SIFT	(II) real-valued	✓/✓/-	512	239	-	-
sRD-SIFT	(II) real-valued	✓/✓/✓	512	-	-	-

Table 5.2. Descriptors

and the neighborhood can be directly observed on the sphere and is independent of the position in the image. They train a FAST detector on the spherical neighborhood representation and dub the detector SFAST (Spherical FAST).

Summarizing, most detectors satisfy the requirements on providing a rotation and scale estimate (Table 5.1). Blob-like detectors, though, remain computationally to expensive for real-time applications. BRISK and ORB, however, are a lot faster than their blob-like competitors by adapting highly efficient corner detectors in image pyramid schemes. Concerning robustness to image distortions and non-perspective projections, the range is limited. The proposed blob detectors are even slower than their predecessors. SFAST could be an alternative, but at the time of writing was not available.

In the remainder of this work, the ORB detector scheme is used to detect keypoints. It is still the fastest method providing a stable rotation estimate. By replacing the FAST detector with AGAST, we are able to gain some additional ms and do not have to retrain the detector to different environments. Yet, the aforementioned drawbacks of using the FAST (or AGAST) detector on fisheye images remains and could be subject to future work.

5.1.2 Keypoint Descriptors

To facilitate the choice of a descriptor for any given task, it is usually advantageous to classify given methods into different categories that are of interest. In [74] the authors propose a taxonomy that puts descriptors into four categories relating the time to compute a descriptor and its storage requirements. Both are also important in the context of large-scale egomotion estimation and we adopt their taxonomy. In the following, we assume that a detector was used to extract keypoints with corresponding orientation and scale information.

Type I - Image patches

The most basic type of descriptor is an image patch surrounding the detected keypoint. The matching of such image patches can then be performed by computing, for example, the sum of squared differences (SSD) or the normalized cross correlation (NCC). Scale, rotation and distortion invariance can be achieved, by rotating, scaling and undistoring the patch content which, however, increases the computational burden per patch. In addition, the storage requirements grow quadratically with the patch size, making this type of descriptor impracticable for large-scale operations.

Type II - Real-valued

The second category contains real-valued descriptors such as SIFT and SURF. By applying a succession of image processing operations such as image derivatives, linear transformations, spatial pooling, histograms, etc. the patch content is mapped to a vector of fixed length reducing the memory footprint. In this category, rotation invariance is achieved by rotating gradient orientations w.r.t a dominant orientation and scale can be compensated by computing the descriptor on the image corresponding to the keypoint scale in the image pyramid. Compared to the type I descriptors, however, the computational effort grows as each patch undergoes the series of image processing steps.

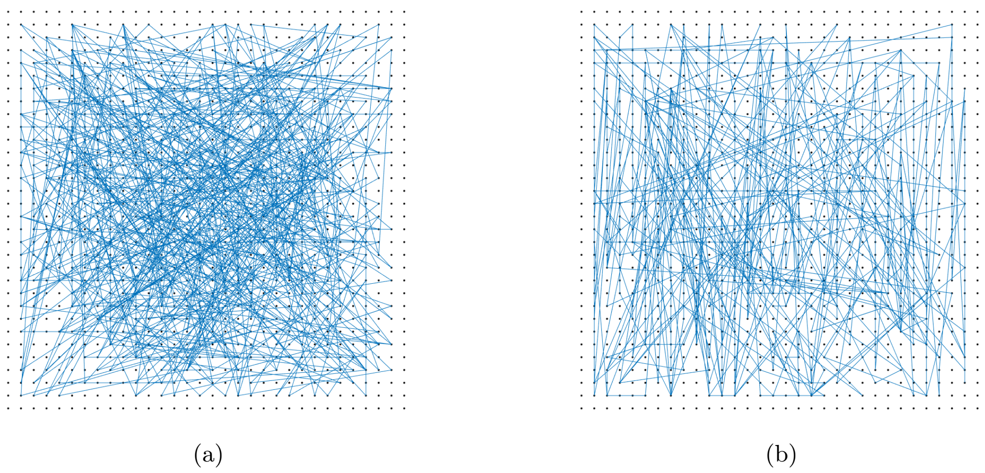


Figure 5.1. Depicted are 256 tests of a test set **Q** (a) The tests are randomly sampled from an isotropic Gaussian distribution, mirroring the BRIEF descriptor. (b) The tests are learned in a scheme similar to ORB. The tests have the property of low correlation and high variance.

Type III - Binarized

The algorithms in the third category try to reduce the memory footprint even further by means of quantizing the real valued descriptors into binary strings (binarization) [24, 205] or by applying dimensionality reduction techniques (PCA-SIFT [82]) without significant loss in matching accuracy. But still, the elaborate pre-processing of the patch content has to be performed and thus the description time for each patch remains high.

Type IV - Binary

This category subsumes so called binary descriptors. Here, each descriptor dimension is the result of a simple binary comparison on pixel intensities in the patch surrounding the keypoint. Thus, the time needed to compute a descriptor drops significantly. This idea was initially proposed by [25] and is called Binary Robust Independent Elementary Features (BRIEF). They define a test τ on a smoothed image patch **P** as follows:

$$\tau(\mathbf{P}; \mathbf{u}_1, \mathbf{u}_2) = \begin{cases} 1, & if I(\mathbf{p}, \mathbf{u}_1) < I(\mathbf{p}, \mathbf{u}_2) \\ 0, & otherwise \end{cases} \tag{5.1}$$

where $I(\mathbf{P}, \mathbf{u})$ is the intensity at pixel location $\mathbf{u} = (u, v)^T$ in the patch **P**. The set **Q** of $d = 1, .., D$ test pairs

$$\mathbf{Q} = \begin{pmatrix} \mathbf{u}_{11}, ..., \mathbf{u}_{1d} \\ \mathbf{u}_{21}, ..., \mathbf{u}_{2d} \end{pmatrix} = (\tau_1, .., \tau_d) \tag{5.2}$$

is fixed and its choice affects the recognition performance. Calonder et al. [25] experimented with different methods to choose the test locations and sampling the test set from an isotropic Gaussian distribution worked best. Figure 5.1a depicts such a set of randomly sampled tests. The size of the descriptor (equivalent to the number of tests) can be arbitrarily chosen. Naturally, a larger number of tests increases the matching performance but the improvement saturates already around 512 tests [25]. The descriptor can be made invariant to rotation and scale by simply transforming (rotating, scaling) the tests around the keypoint (patch center). Unfortunately, the rotation of the test pattern decreases the variance, which makes the feature less discriminative [154]. In addition, the authors of ORB show that tests can be correlated, i.e. some tests do not contribute equally to the results. They propose an unsupervised learning scheme to find the tests (of all possible tests in a patch) that have the highest variance, whilst being uncorrelated. Such a set of learned tests is depicted in Figure 5.1b. We will come back to this training scheme in the next section. Instead of sampling tests randomly or learning uncorrelated test, the BRISK [106] descriptor is build from a set of equally spaced tests that are located on circles concentric with the keypoint. As mentioned in the detector section, the patch orientation is estimated using the gradients from the long-distance pairs, while the short-distance pairs are used to compute the descriptor.

Learned descriptors

Thus far, most descriptors are ”hand-crafted”, i.e. the image processing operations within the patch, as well as their order, were carefully selected and tuned. Large ground-truth datasets of matching and non-matching patches [205] allowed to take a different approach and involve machine learning to map the patch content to a discriminative descriptor. In [205] optimal descriptor parameters for the DAISY [188] descriptor are learned. In [104] the image processing operations (called building blocks), as well as their order is learned. To create the BinBoost [191] descriptor, the authors learn a hash function that directly maps patch gradient responses to a binary string. Although the descriptors of those methods are the results of learning parts of the processing chain, the authors still selected which building blocks to consider. The recent success of Convolutional Neural Network (CNN)s and their ability to learn which features and representations are relevant to minimize a given cost function, motivated the authors of [66, 170] to learn not only the ”patch to descriptor” mapping as a whole [12], but also distance metric between two descriptors.

Still, the mapping from patch to descriptor is learned off-line from large datasets of patches, that originate from scenes representing distinct and fixed characteristic. An intuitive improvement is to adapt the descriptor to new scenes. Unfortunately, the off-line training involved in most methods is cumbersome and requires (up to) days for training. Recent methods [13, 150] propose to adapt the BRIEF descriptor presented above for every patch or point, respectively. The basic idea is to learn a boolean mask of the same length as the fixed binary test set $\mathbf{Q}$ (Equation 5.2), that suppresses tests that have a varying outcome for different views and are thus unstable. This mask can be found by transforming the fixed tests $\mathbf{Q}$ on-line and reapply it to the patch. The resulting descriptors should be equal, if all tests have the same outcome. If not, the corresponding mask entry is set to 0, i.e. the test will be suppressed during matching. For TailoredBRIEF [150] the authors propose to sample many affine views of the tests for reference points online. Then, while matching descriptors from the current camera image to descriptors from previously extracted landmark points, the learned mask is applied to discard unstable tests. They mention that masks for both the reference and the current descriptors could be learned, but argue, that for real-time tracking systems this is not necessary. For the Binary Online Learned Descriptor (BOLD) [13], the authors point out the similarities of learning masks to Linear Discriminant Analysis (LDA) based approaches [179]. In principal, LDA tries to find a linear combination of features, that best separates two classes (matching, non-matching points), by maximizing inter-class (different patch) and minimizing intra-class (same patch) variance. Learning uncorrelated tests off-line, like done for the ORB descriptor, increases the inter-class distance by maximizing the variance across inter-class patches. Then, by sampling and finding stable tests online, the intra-class variance is minimized. For BOLD, masks for all patches are learned and applied during matching. We will embrace both ideas (off-line and on-line learning), discuss them in more detail and adapt them to the task of this work in the next section.

Lets recall the demands on the descriptor for the task of multi-fisheye camera egomotion estimation. Apart from being rotation and scale invariant, the descriptor should be binary, as the storage requirements as well as the matching speed significantly outperform their real-valued counterparts. Methods that binarize real-valued descriptors (type III) or learn a mapping from patch to descriptor are still to slow for mobile real-time applications. This leaves us with binary descriptors that are created from binary intensity comparisons. In addition, the online adaption of such binary descriptors is possible [13, 150]. Ultimately, the question remains how to adapt the binary descriptors to image distortion or non-perspective projections. Most state-of-the-art descriptors assume that the patch content surrounding a keypoint originated from a perspective projection and can be locally approximated by an affine transformation [131]. Most commonly the lens distortion is ignored, as it is assumed that images are undistorted before feature extraction is performed. As discussed above, the undistortion of wide-angle, fisheye or omnidirectional images often leads to re-sampling artifacts due to interpolation.

In the following section, we will take a closer look at combining the advantages of learning BRIEF-like tests off-line (ORB) and on-line (BOLD) and applying image distortion to the tests instead of the patches, which is a lot faster to compute.

5.2 dBRIEF - Distorted BRIEF

At the beginning of this section a modified (distorted) version of the BRIEF descriptor is presented. We then show in simulations how the modifications transfer to Hamming distance distributions be-

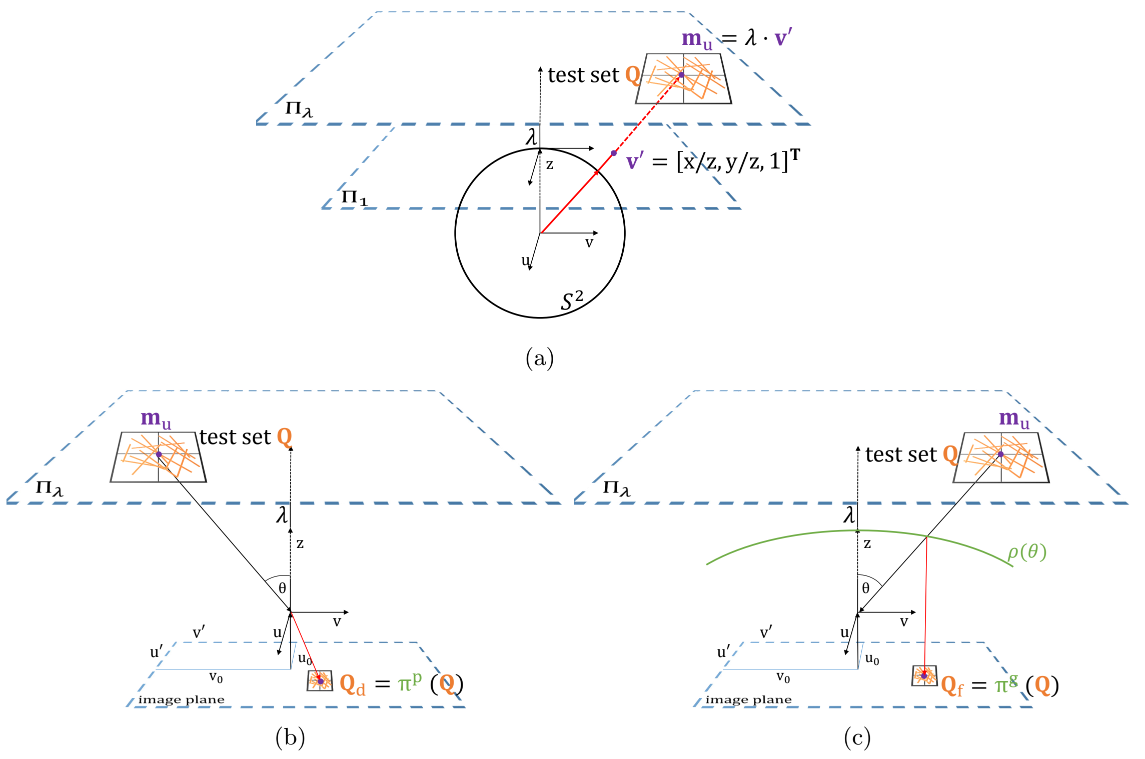


Figure 5.2. (a) Undistortion of the current image point. After projecting the point to the unit sphere S^2 and subsequent projection onto the normalized image plane Π_1 the point $\mathbf{v}^{'}$ is scaled to a plane Π_λ that lies in distance λ from Π_1. Then the test set $\mathbf{Q}$ is added to the point coordinates. (b) and (c) show the distortion of the test set $\mathbf{Q}$ to its corresponding distorted test set $\mathbf{Q}_d$ and $\mathbf{Q}_f$ for a simple radial distortion model and a fisheye projection respectively.

tween matching and non-matching descriptors and the recognition rate respectively. Subsequently, an optimal, low correlation test set is learned, following the unsupervised learning scheme presented in [154]. We contribute an additional study on how the learned test set changes if different detectors are used to acquire the initial set of keypoints for learning. Further, we investigate if training on fisheye images improves performance. Following the off-line learning, we present the online mask learning for our distorted descriptor and evaluate the developed methodology against state-of-the-art methods w.r.t speed and matching performance on real data.

To show both the effect of radial distortion as well as non-perspective mappings on the BRIEF descriptor, we first introduce the simple radial distortion model [114] with one parameter ξ controlling the amount of radial distortion:

$$\mathbf{m}_d = r(\mathbf{m}) = \frac{2\mathbf{m}}{1 + \sqrt{1 - 4\xi\|\mathbf{m}\|^2}} \tag{5.3}$$

Again, $\mathbf{m} = [u, v]^T$ is an image point in sensor coordinates and $\mathbf{m}_d$ its distorted counterpart. Note, that we chose this model for the sake of simplicity and speed, as it is analytically invertible. This model could be exchanged with every other distortion model.

From Equation 5.3 it is obvious that the distorted coordinates $\mathbf{m}_d$ depend on the undistorted point $\mathbf{m}$. In order to calculate a distorted version of our test set $\mathbf{Q}$, we thus have to get the coordinates of the undistorted points first. Let $\mathbf{m}'$ be a keypoint detected in the distorted image. Assume an arbitrary interior orientation parametrization π including Equation 5.3 then the projection of $\mathbf{m}'$ to its corresponding (undistorted) bearing vector is:

$$\mathbf{v} = [v_x, v_y, v_z] = \pi^{-1}(\mathbf{m}') \tag{5.4}$$

and its counterpart on the normalized image plane Π_1:

$$\mathbf{v}' = [v_x/v_z, v_y/v_z, 1] \tag{5.5}$$

This transformation is depicted in Figure 5.2a. Now, $\mathbf{v}^{'}$ can be scaled to a plane Π_λ with distance λ from the normalized image plane, yielding

$$\mathbf{m}_u = [m_x, m_y, m_z]^T = \lambda \mathbf{v}' \tag{5.6}$$

For a perspective camera λ would equal the focal length, for the omnidirectional model, we set $\lambda = a_0$, i.e. the first coefficient of the forward projection polynomial. Subsequently, the fixed test set $\mathbf{Q}$ can be anchored on $\mathbf{m}_u$.

$$\mathbf{Q}_u = \mathbf{Q} + \mathbf{m}_u \tag{5.7}$$

The final distorted version of the test set $\mathbf{Q}_d$ is then given by back-projection of $\mathbf{Q}_u$ onto the image plane. Three realizations, of this projection are given by the following equations:

$$\text{perspective: } \mathbf{Q} = \pi^p(\mathbf{p}_i) = \frac{\mathbf{Q}_u}{m_z} + \mathbf{o} \tag{5.8}$$

$$\text{perspective + distortion: } \mathbf{Q}_d = r(\frac{\mathbf{Q}_u}{m_z}) + \mathbf{o} \tag{5.9}$$

$$\text{fisheye: } \mathbf{Q}_f = \pi^g(\mathbf{Q}_u) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & 1 \end{bmatrix} \begin{bmatrix} \rho(\theta) m_x / r \\ \rho(\theta) m_y / r \end{bmatrix} + \begin{bmatrix} o_u \\ o_v \end{bmatrix} \tag{5.10}$$

with $r = \sqrt{m_x^2 + m_y^2}$ and $\theta = \arctan(m_z/r)$. The first version (Equation 5.8) equals the standard version of the descriptor generation. In the second version (Equation 5.9), radial distortion is applied to the tests in the sensor plane using Equation 5.3. This projection is depicted in Figure 5.2b. The third version is the non-perspective projection of the test set. We use the generic camera model introduced in Section 3.2, but other projection models could be used as well [162].

5.3 Simulations

To show the validity, performance and properties of our distorted versions of BRIEF (Equation 5.9 and Equation 5.10), we perform a simulation and various tests. We simulate a projection of the first image of the popular Graffiti dataset [127] to a virtual camera. Therefore, the image is mapped to the x-y plane in 3D. Then, a virtual camera with known exterior orientation $\mathbf{M}$ and fixed image size moves along the x-axis and observes the Graffiti plane. Three versions of this sequence are depicted in Figure 5.3. The images in each column originate from the same exterior orientation. The rows represent different interior orientation parametrization. For the first row, a standard perspective projection is used. This simulation is used as the source implementation of our distorted versions, as its outcome is supposed to be predictable and equals the behavior of BRIEF. The second row mirrors a perspective projection with radial distortion (Equation 5.9). We set the distortion coefficient to $\xi = -2 \cdot 10^{-6}$ for all experiments. The third row depicts a virtual fisheye camera. We use the real calibration data from one of the fisheye cameras used in this thesis.

To study the transformation of the original test set $\mathbf{Q}$, a keypoint is picked on the original Graffiti image. Then, we select a patch of fixed size (32x32 pixel in all experiments), around its projection in the virtual camera (red squares in Figure 5.3). Figure 5.4 depicts smoothed versions (Gaussian kernel of width 2) of the extracted patches from the image sequences. The leftmost column shows the original patch from the Graffiti image. Obviously, the original patch content gets significantly distorted and compressed except, of course, for the standard perspective projection. This gives rise to two observations. First, assuming the detector manages to extract the same keypoint over the sequence, the patch content is highly dependent on the position in the image. Second, the application of the standard test set $\mathbf{Q}$ on the patch will probably have a different outcome (and thus descriptor) for each patch of the distorted versions. In addition, this difference will be bigger the farther we move away from the original patch (original camera pose). This, of course, is unfavorable, as matching performance will drop significantly as the Hamming distance between the same descriptor grows.

To visualize the second observation, we overlay the original test set $\mathbf{Q}$ and the transformed versions of it on some patches of the sequence. This is depicted in Figure 5.5. The red lines depict the first 20 tests from the original test set $\mathbf{Q}$. The green lines depict the first 20 tests from the distorted test sets $\mathbf{Q}_d$ and $\mathbf{Q}_f$ in the middle and last row respectively.

Evidently, the outcome of testing pairwise pixel intensities with the distorted tests will be different. To analyze the magnitude of this difference, we extract the descriptor for each patch of the sequence

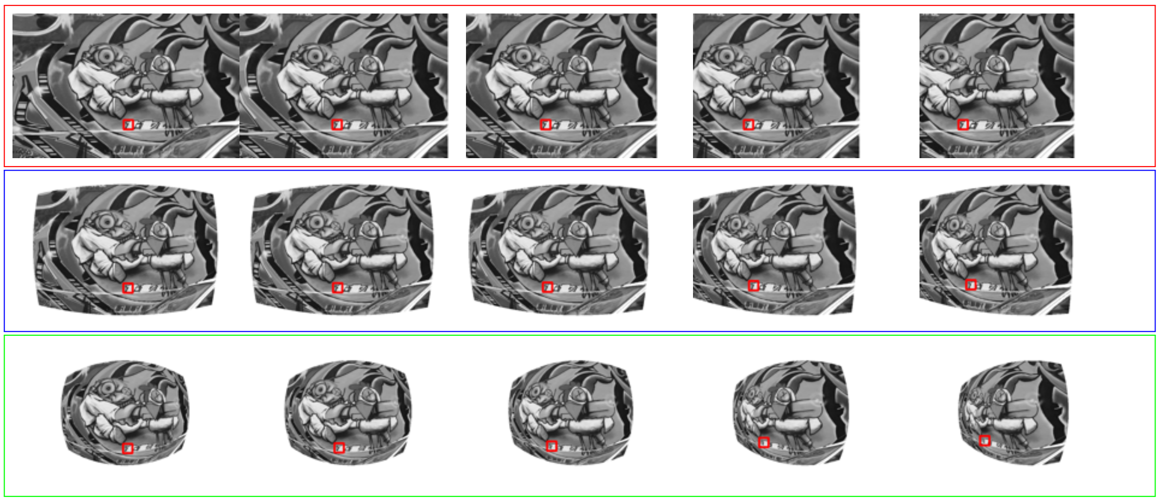


Figure 5.3. Tracking simulation. Camera is moving in x-direction, i.e. from left to right in this simulation. A patch (small red square) is tracked in each image. Red box: perspective projection. Blue box: added radial distortion ($\xi = -2 \cdot 10^{-6}$). Green box: fisheye projection.

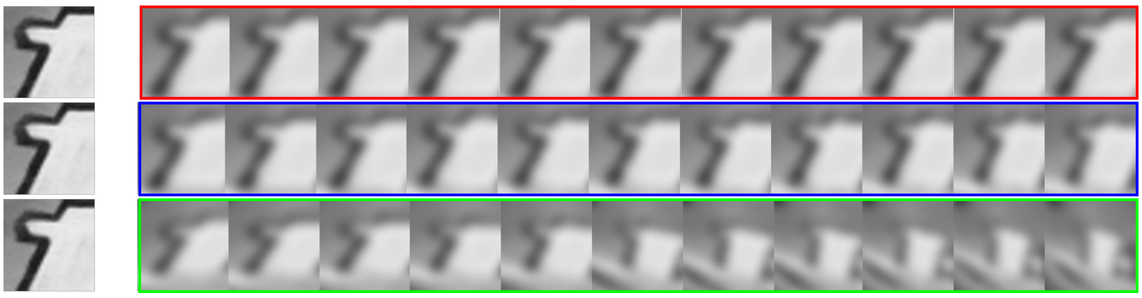


Figure 5.4. Patch (red square in Figure 5.3) for each of the three cases. Again: red for perspective projection, blue for radial distortion, green for fisheye projection.

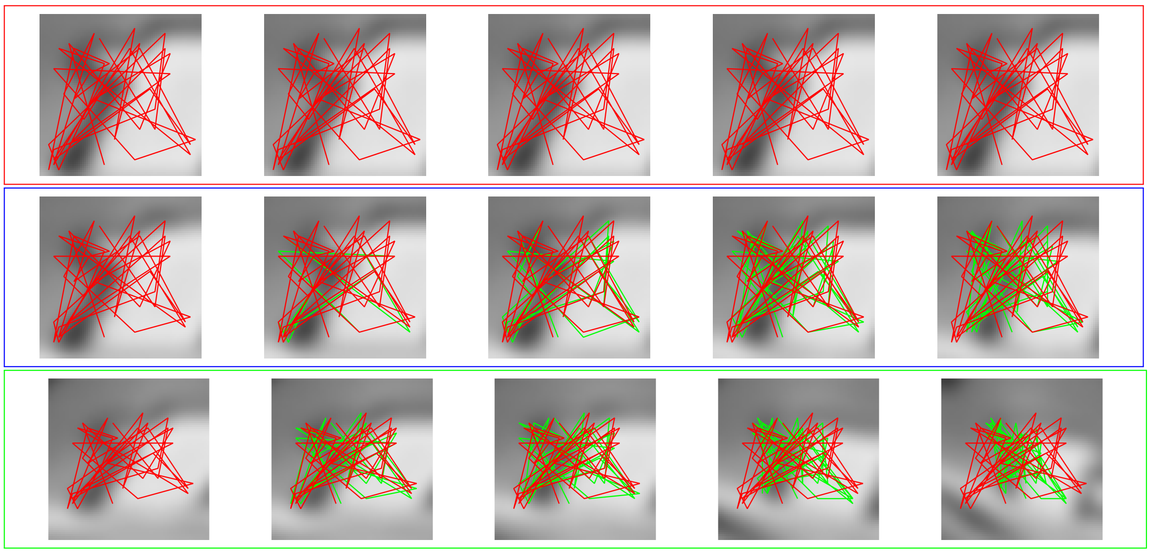


Figure 5.5. Effect on the test set $\mathbf{Q}$. Red: original tests $\mathbf{Q}$, Green: distorted tests $\mathbf{Q}_d$ and $\mathbf{Q}_f$ in second and third row respectively.

using $\mathbf{Q}$ and $\mathbf{Q}_d, \mathbf{Q}_f$ respectively. Then we compute the Hamming distances between the first and each subsequent descriptor of the sequence. In general, the Hamming distance between two binary descriptors d_i and d_j is given by:

$$H(\mathbf{d}_i, \mathbf{d}_j) = \mathbf{d}_i \oplus \mathbf{d}_j \tag{5.11}$$

where $\oplus$ is the logical XOR operator. Ideally, these distances would be zero, which, however, is unrealistic in practice, due to noise, discretization, illumination and viewpoint changes. Practically, the Hamming distance between the descriptors of equal patches should be small and constant over the sequence. In other words, the intra-class distance as well as its variance should be as small as possible.

The evolution of the Hamming distance for standard BRIEF ($\mathbf{Q}$) as well as descriptors extracted using $\mathbf{Q}_d$ and $\mathbf{Q}_f$ is depicted in Figure 5.6a and Figure 5.6b respectively. Applying the original test set $\mathbf{Q}$ yields the red graphs (BRIEF), whereas the blue graphs represent the Hamming distance of a distorted version of BRIEF, which we abbreviate with dBRIEF (not to be confused with D-BRIEF [193]). For the radial distortion version of dBRIEF (Figure 5.6a), the Hamming distance stays quite constant and is below 30 bits over the whole sequence. In contrast, the distance between the first and last descriptor of the sequence for the BRIEF descriptor grows continuously and reaches 70 bits which is about a factor of 2.3 larger. Figure 5.6b shows a similar behavior and BRIEF even reaches 110 bits. The dBRIEF version for the fisheye camera stays quite constant in the beginning, but then the distance starts to increase around the 20th image and reaches 60 bits. Still, the error is about two times lower, which is expected to be advantageous during matching.

To test the matching performance of dBRIEF, we first extract 200 keypoints on the original Graffiti image, project their location throughout the image sequences and subsequently extract a descriptor for each patch using BRIEF and dBRIEF respectively. The patch locations are depicted in Figure 5.7 on three images of the sequence (10 views in total). We measure the matching performance using the recognition rate (or recall). This rate shows the ability of the descriptor to produce correct matches.

Let $\mathbf{D}^i = (\mathbf{d}_1^i, .., \mathbf{d}_N^i)$ be a set of N descriptors extracted in the i-th image and C the coincident keypoints detected in both images. Usually $C = \mathbf{D}^i \cap \mathbf{D}^j$, but as we do not detect keypoints in every frame but rather project them using the known orientation, $C = \mathbf{D}^i = \mathbf{D}^j$, i.e. all keypoints appear in all images and we know that each keypoint has a match. In the particular simulation $N = 200$. The set of correct matches S_{true} is found by matching each descriptor in the i-th image with each descriptor from the j-th image. A match is identified as correct *iff* the distance between two descriptors is minimal. The recognition rate is computed as follows:

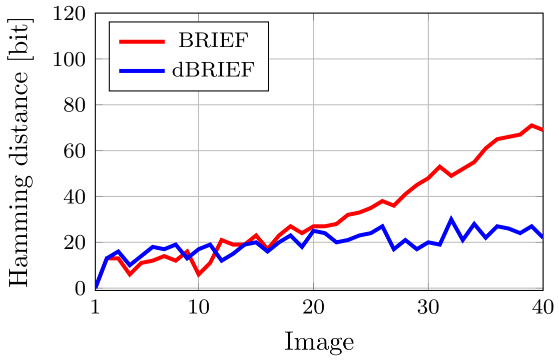
$$\text{recognition rate}(t) = \frac{\#S_{true}}{\#C} \tag{5.12}$$

where $\#$ is the count and t a threshold on the descriptor distance. We match the descriptors of each image of the sequence to the set of descriptors from the first image. The threshold t for this experiment is not set, i.e. each descriptor will have a corresponding match and $\#C = 200$.

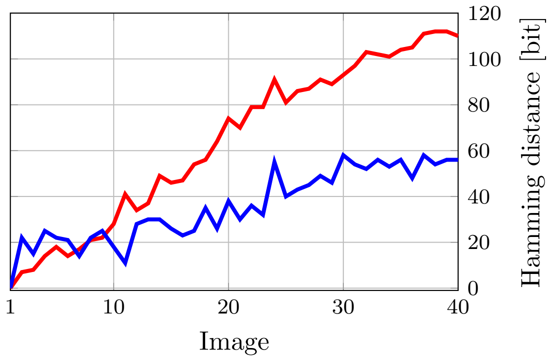
The recognition rates are depicted in Figure 5.8. Figure 5.8a shows the results for the radial distortion version of dBRIEF ($\mathbf{Q}_d$) and Figure 5.8b the fisheye version of dBRIEF ($\mathbf{Q}_f$). In both cases dBRIEF outperforms BRIEF. In the radial distortion case, the recognition rate for BRIEF systematically drops from 100% to about 60% and in the fisheye case down to 40%. The dBRIEF descriptors return more correct matches and thus the recognition rates remains above 70% and around 60% for radial distortion and fisheye images respectively. This signifies a relative improvement of 20% and 50% respectively.

To get a better understanding of the improvement, we visualize the relative frequencies of Hamming distances of matching and non-matching descriptors in Figure 5.9. The blue and red distributions represent the frequencies of matching and non-matching descriptors respectively. If the distributions do not overlap, only correct matches would be found. The ultimate goal of improving matching performance for descriptors is to get the distribution overlap as small as possible by decreasing the intra-class distance (making the distributions as tight as possible) and by increasing inter-class distances (distance between the mean of each distribution).

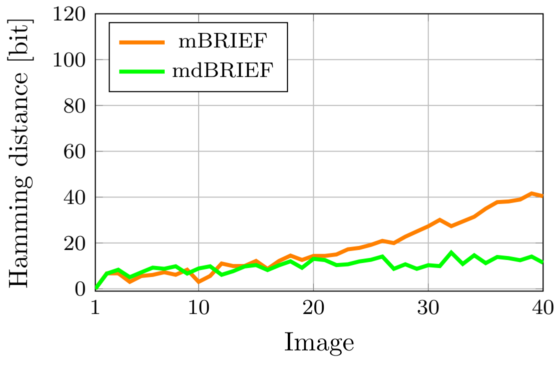
The first two rows of Figure 5.9 visualize the distributions for every third pair of the fisheye sequence. Mean and variance of the red distribution (non-matching points) remain quite constant.



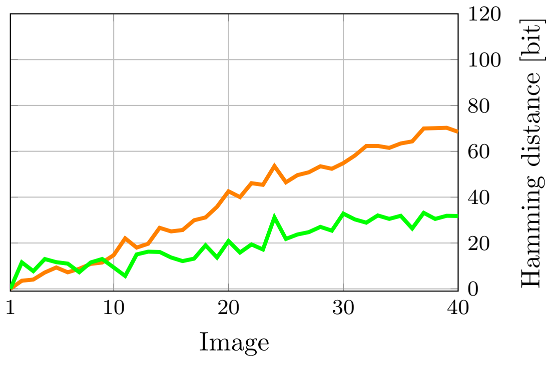
(a) radial distortion



(b) fisheye projection



(c) radial distortion



(d) fisheye projection

Figure 5.6. Evolution of the Hamming distance for the same patch across the simulation sequence but different test sets. (a)-(b) depicts the evolution for the original BRIEF and the distorted version dBRIEF. (a) radial distortion simulation (cf. blue box Figure 5.3) and (b) fisheye projection simulation (cf. red box Figure 5.3). In contrast (c)-(d) depicts the masked version of BRIEF and dBRIEF descriptors for the radial distortion and the fisheye projection simulation respectively.

The mean of the blue distribution (matching points), however, starts to shift towards the mean of the red distribution. In addition the variance increases, i.e. the distribution gets broader. This can be directly associated with the drop in recognition rate. To quantify the improvement of dBRIEF over BRIEF, we calculate the Bhattacharyya coefficient c_{ba}, which is an approximate measure of the overlap of two distributions. For the last image pair (1 to 10), the coefficient for BRIEF and dBRIEF is 0.5254 and 0.3947 respectively (27% lower).

5.4 Improving dBRIEF by Online Mask Learning

As mentioned frequently throughout this chapter, one way of improving the matching performance is to decrease the intra-class variance whilst simultaneously increasing inter-class distance between matching and non-matching descriptors. Most learning methods [66, 104, 170, 179, 191–193]) optimize both criteria offline and simultaneously for large datasets of corresponding patches. To minimize the intra-class variance, patches of the same keypoint are needed. Even though the datasets provide multiple instances of the same patch, the optimization of the intra-class variance is performed over descriptors that originate from the same image category and contain similar image characteristics and patch content. As pointed out by [13], an online adaption of the descriptor creation, i.e. a per patch optimization of the intra-class variance for a specific problem/camera/scene should be beneficial and increase matching performance.

This is, however, computationally too expensive for sophisticated, yet cumbersome optimization algorithms that learn functions globally that map the patch content to descriptors. Thus, we will follow the ideas of [13, 150, 154] and adapt the binary descriptors online. Fortunately, this is not too expensive to compute for binary descriptors and exploits the properties of binary tests, briefly resumed in the following.

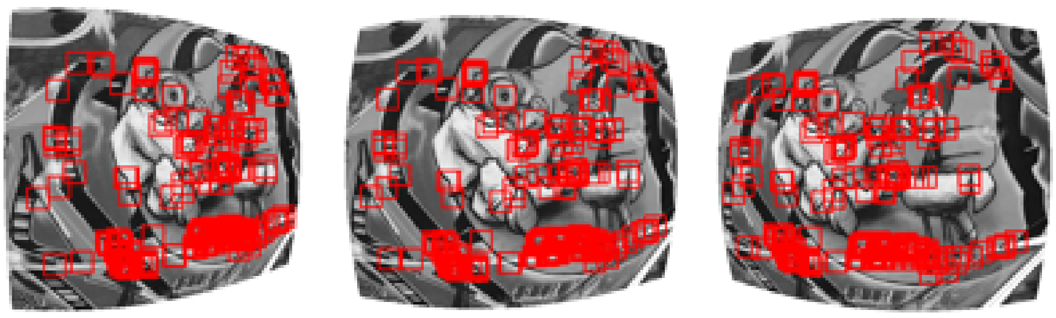


Figure 5.7. Images with 200 tracked ORB features. Depicted are image 1, 7 and 10 from the radial distortion image set. Fisheye images are not displayed here.

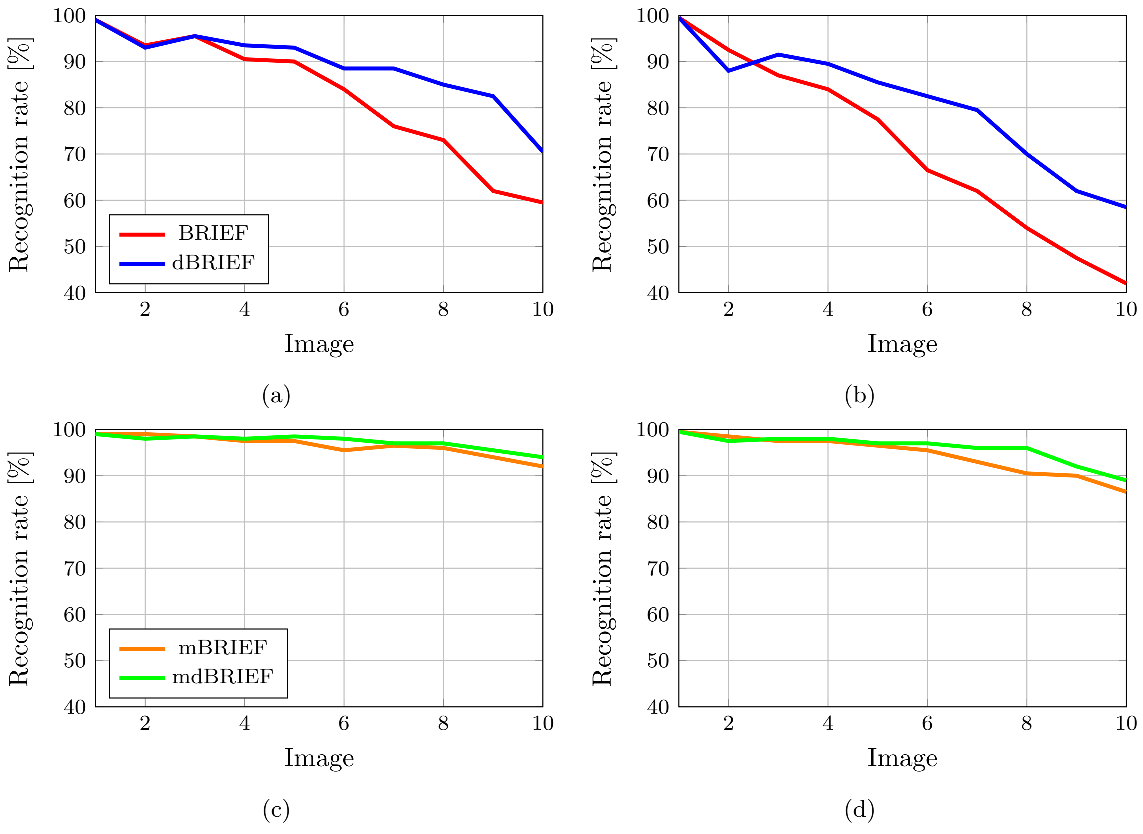


Figure 5.8. Recognition rates for 200 ORB features. In case of the masked version masks are learned for all patches and the Hamming distance is computed with two masks. (a)-(b) BRIEF and dBRIEF for radial distortion and fisheye projection respectively. (c)-(d) masked versions for radial distortion and fisheye projection respectively.

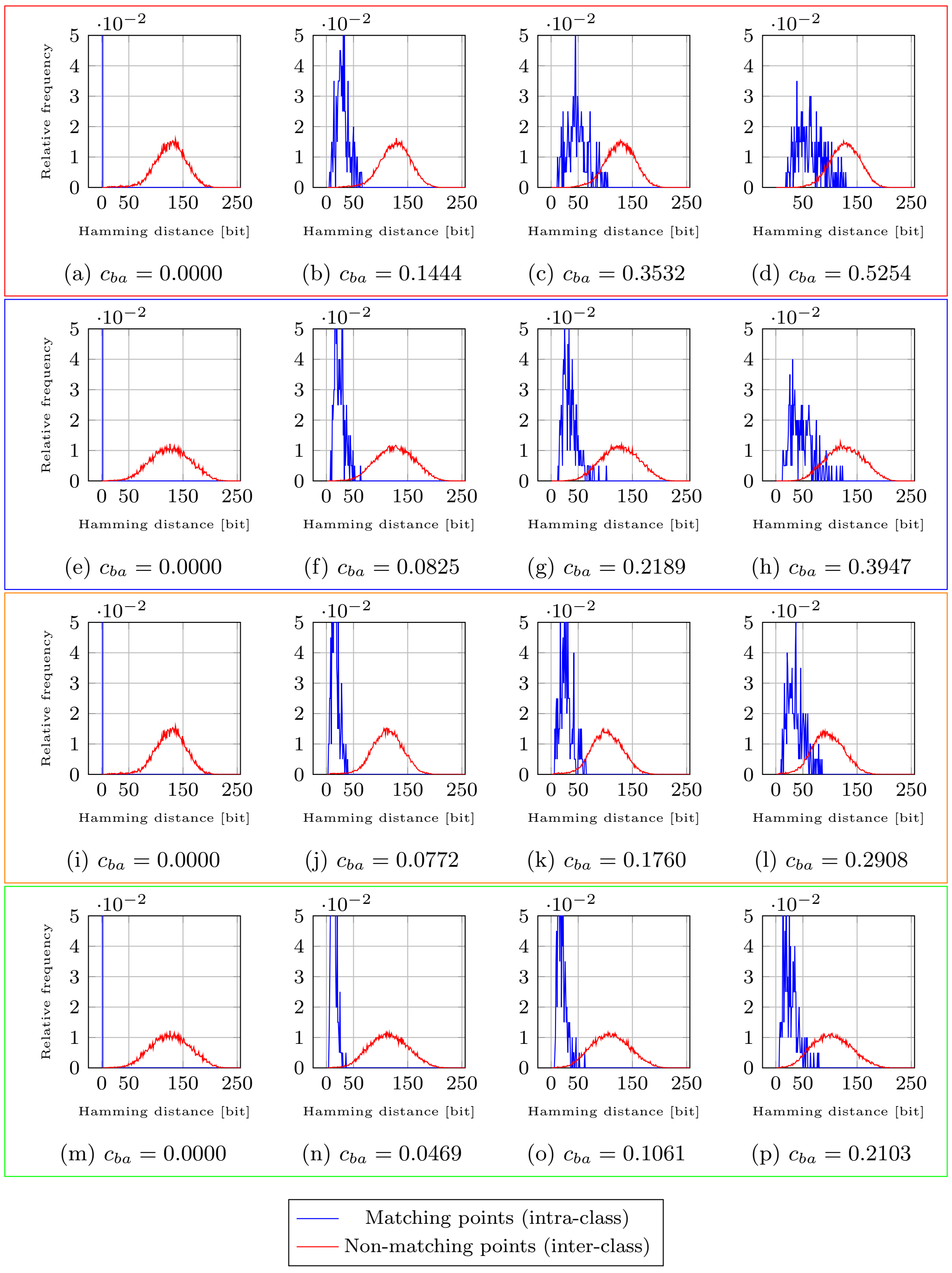
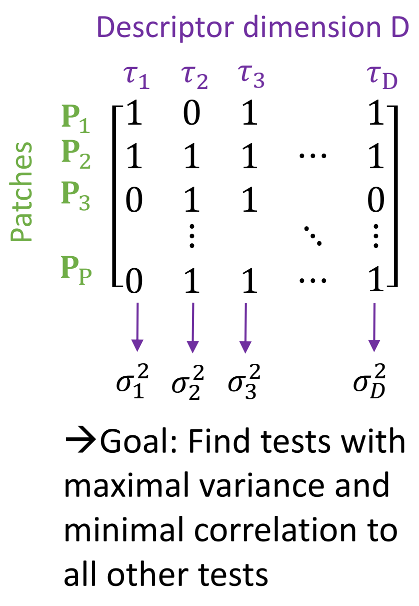
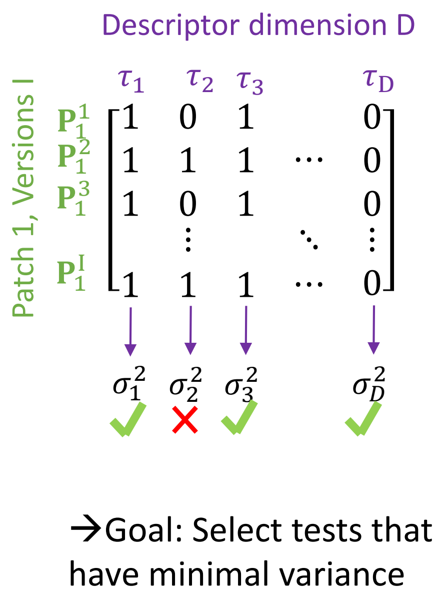


Figure 5.9. Hamming distance distributions for matching (intra-class) and non-matching (inter-class) binary descriptors for the fisheye camera simulation. Each column represents the distributions for the image pairs 1-1,1-3,1-7 and 1-10 respectively. c_{ba} is the Bhattacharyya coefficient for the corresponding plot. First row (a)-(d) BRIEF, second row (e)-(h) dBRIEF ($\mathbf{S}_f$), third row (i)-(l) mBRIEF, fourth row (m)-(p) mdBRIEF.


Descriptor dimension D
Patches
τ_1 τ_2 τ_3 τ_D
P_1 P_2 P_3 P_P
σ_1^2 σ_2^2 σ_3^2 σ_D^2
→Goal: Find tests with maximal variance and minimal correlation to all other tests

(a) Inter-class optimization


Descriptor dimension D
Patch 1, Versions I
τ_1 τ_2 τ_3 τ_D
P_1^1 P_1^2 P_1^3 P_1^I
σ_1^2 σ_2^2 σ_3^2 σ_D^2
→Goal: Select tests that have minimal variance

(b) Intra-class optimization

Figure 5.10. (a) The matrix has dimension $P \times D$. The goal is to increase the inter-class distances. Each column represents one test τ of the descriptor and each row represents the results of applying τ to patch $\mathbf{P}$. Thus a variance σ^2 of one particular test over all patches can be computed. (b) The matrix has dimension $I \times D$. To minimize intra-class distances tests that have zero variance across different versions of the same patch are selected.

In general, the outcome of a binary test is described by a Bernoulli distribution, given the probability α of success (value 1) and $\beta = 1 - \alpha$ failure (value 0). The mean of a binary test is given by α and the variance by $\sigma^2 = \alpha(1 - \alpha)$.

Now, let $\mathbf{A}$ be a $P \times D$ matrix of P binary descriptors of dimension D. This matrix is depicted in Figure 5.10a. Each column of $\mathbf{A}$ represents one binary test (descriptor dimension). The outcome of this binary test τ_d on a particular patch $\mathbf{P}$ are stored in the rows of $\mathbf{A}$ at position $a_{p,d}$. To learn a globally best set of binary tests, the variance $\sigma_d^2 = \alpha_d(1 - \alpha_d)$ of a particular test τ_d should be high [154], where $\alpha_d = \sum_P a_{p,d}/P$ is the ratio of ones for test τ_d on a patch $\mathbf{P}$. This is reasonable, as the outcome should be as variable as possible on different patches (represented by the rows of $\mathbf{A}$). To learn a locally best set of binary tests, the rows are replaced by different instances/representations of the same patch. This can either be achieved by warping the patch using small rotations or affine deformations or by applying the deformations to the test set. Then, the goal is to find those tests, that have the lowest variance (preferably zero), over all instances of the patch. This is depicted in Figure 5.10b. In this example, test two has non-zero variance and is suppressed.

In the following two sections, we briefly describe the offline and online learning for dBRIEF and subsequently present results on simulated and real data.

5.4.1 Offline Learning

We follow the unsupervised learning of a set of high variance tests with low correlation which was first presented in [154]. Since then, various authors [4, 13, 107] adopted the learning scheme. Very recently, the authors of BinBoost [191] proposed a new learning method for intensity test configurations [190] by generalization of the employed boosting method for gradient based descriptors. Although they achieve a performance gain over the greedy approach of [154], we use latter to learn our test set off-line, and postpon ea comparison to [190] to future work.

Given a patch of size $S \times S$, the number of possible intensity comparisons within this patch is $D = \binom{S^2}{2}$. Such a binary pixel intensity test is defined as in Equation 5.1 and we construct a set $\mathbf{Q}_{all}$ of all possible test configurations. For a patch size of $S = 32$, that is used for all experiments, the total number of all tests would be 523776. To reduce this number, we omit tests along the patch border and tests that have a distance $d < 3$ or $d > 9S/10$, leaving us with a total number of 377650 possible binary tests. Then, we extract 20000 patches from two different datasets. The first dataset consists of images from the PASCAL visual object classes challenge [48] and is the same as used in [154]. The second is a fisheye image dataset. To obtain various types of fisheye images, we performed a search for fisheye images and randomly downloaded 300 images.

In addition the extracted patches are computed from keypoints obtained by three different detectors. We chose ORB, AKAZE and SURF to detect keypoints on both datasets. All three descriptors use different methods to detect a keypoint and compute its dominant orientation. The loss in test variance in the original BRIEF descriptor comes basically from rotating the test set according to the keypoint orientation [154]. During learning, each test is rotated before applying it to the patch, thus the learned results will depend on the estimated rotation. This study is motivated by the fact that none of the papers that adopted this greedy learning scheme investigated the impact of using different images and detectors on the overall performance of the final descriptor. The fact that the BOLD descriptor is learned on a different dataset than ORB, but has a better performance (without masks) indicates, that this influences the result.

The off-line learning is conducted in two steps. First, each test is performed on each patch. In theory, this results in a matrix $\mathbf{A}$ of size $P \times D$. As this matrix would not fit into memory, we directly compute the variance σ_d^2 for one test τ_d applied to all patches. Then, we sort the D binary tests according to their variance and save the sorted test set to $\mathbf{Q}_{sorted} = (\tau_1^s, .., \tau_D^s)$.

In the second step, we try to identify tests that have a correlation $c < t_c$, where t_c is a threshold. In order to find such low correlation tests, we start with the first test from $\mathbf{Q}_{sorted}$ and save it to the final test set $\mathbf{Q} = (\tau_1, .., \tau_D)$. Then, we iterate through all patches, apply one sorted test τ_d^s and compare the correlation to all tests from the final test set $\mathbf{Q}$. The correlation is computed as [13]:

$$c = |\frac{2}{P} \sum_{p=1}^{P} |\tau_{d,p} - \tau_{d,p}^s| - 1| \tag{5.13}$$

This value thus varies between 0 and 1 where 0 is no and 1 perfect correlation, respectively. We start with a correlation threshold of $t_c = 0.2$ and increase this value by 0.1 every time we iterated over all tests $\mathbf{Q}_{sorted}$. This is conducted until a desired number of tests is found (e.g. 512) that yield the final descriptor.

5.4.2 Online Mask Learning

After having identified a high variance-low correlation test set that maximizes inter-class distance, we can proceed by trying to reduce the intra-class variance per patch. The theoretical background is identical to the one proposed in [13]. Our contribution is to learn masks for our distorted descriptor dBRIEF.

Instead of learning a new test set $\mathbf{Q}$ for each patch (which would be infeasible in practice), we learn a mask $\mathbf{L} = (l_1, ..., l_d)$ of length D, with $l_d \in \{0, 1\}$. This mask suppresses individual tests during matching if $l_d = 0$, i.e. the result of a particular test d has no influence on the Hamming distance between two descriptors.

To learn such a mask online, we find the tests that are stable under various transformations of the original patch. In other words, we try to find tests that have zero intra-class variance. Instead of re-sampling the patch, the off-line learned test set $\mathbf{Q}$ is transformed. Those transformations have to be carried out online, with every extraction of a new keypoint. Luckily, as pointed out by [13, 23] and confirmed by own experiments, two small random rotations work better than performing many full affine transformations.

Now, if a keypoint is detected, we anchor the test set $\mathbf{Q}$ to the undistorted location of the keypoint using Equation 5.7. Here, we perform two small rotations of the test set and project the three test sets back to the image yielding $\mathbf{Q}_d^{0,1,2}$ or $\mathbf{Q}_f^{0,1,2}$ respectively, where the superscript 0 indicates the original test set and 1,2 are the rotated versions respectively. Subsequently, zero variance tests are identified by calculating the variance between each test of $\mathbf{Q}^0$ and $\mathbf{Q}^{1,2}$. This is schematically depicted in Figure 5.10b.

During matching a masked Hamming distance has to be computed. Given two descriptors d_i and d_j with corresponding learned masks l_i and l_j the distance is computed as:

$$H_{masked}(d_i, d_j, l_i, l_j) = \frac{1}{o_i} l_i \wedge d_i \oplus d_j + \frac{1}{o_j} l_j \wedge d_i \oplus d_j \tag{5.14}$$

where o_i, o_j represent the total number of ones in the masks respectively and $\oplus$ is the logical XOR operator. The masking is achieved by the logical AND $\wedge$ operator that suppresses the contribution of a particular test to the Hamming distance.

To compare the performance improvements over the standard BRIEF and dBRIEF descriptor, we analyze the masked version with the same simulation used for the dBRIEF descriptor. The Hamming distance error for one patch of the simulation over the entire sequence is depicted in Figure 5.6. The orange and green graphs in Figure 5.6c and Figure 5.6d depict in the masked version of BRIEF and dBRIEF respectively. Clearly, masking unstable tests for each patch decreases the Hamming distance between the descriptors of the same patch, and thus the intra-class variance. We repeat the experiment for 200 keypoints and calculate the recognition rate. The results are depicted in Figure 5.8c and Figure 5.8d. The masked versions outperform BRIEF and dBRIEF w.r.t the recognition rate. This effect can be further investigated by plotting the Hamming distance distributions for matching and non-matching descriptors and calculating the Bhattacharyya coefficient between them. This is depicted in the last two rows of Figure 5.9. The overlap between the distributions for image pair 1-10 for dBRIEF and masked dBRIEF (last plot in red and green box) is $c_{ba} = 0.3947$ and $c_{ba} = 0.2103$ respectively and thus the overlap of dBRIEF is about 85% higher. In general, the intra-class distributions are a lot tighter for the masked versions, showing the effect of reducing the intra-class variance, by masking non-zero variance tests. In the remainder of this work, the masked version of **dBRIEF** is called **mdBRIEF**.

5.5 Results on Real Data

In this Section, the performance of dBRIEF and mdBRIEF is evaluated against the state-of-the-art on a real data sets. The results for BOLD are based on the code provided by the authors. For all other descriptors and detectors we use the standard implementations of all methods that are part of OpenCV[1].

To evaluate the matching performance, we compute the recognition rate and '1-precision' curves according to [114, 127]. The recognition rate (recall) was already defined in Section 5.3 and measures the descriptors capability of identifying matches from the total set of available points. This measure is complemented by the correctness of those detected correspondences and is expressed in terms of the precision. Let $\mathbf{D}^i, \mathbf{D}^j$ be a set of descriptors extracted from patches around keypoints detected in image i and j respectively. Then the coincident keypoints, i.e. points that are inside the image border and detected in both images are found by projecting the points from image i to image j using a given ground truth transformation (e.g. homography for planar scenes) and taking the nearest neighbor in a radius of 3 pixels. This yields the set of ground truth matching points $C = \mathbf{D}^i \cap \mathbf{D}^j$.

Then we perform Brute Force matching of descriptors, with an increasing Hamming distance threshold t, i.e. a match is identified as correct *iff* the distance between two descriptors is minimal and below t. The set of all matches identified by the algorithm is then given by V. From this set the correct S_{true} and incorrect matches can be identified using the ground truth set C. The precision and recognition rate are then given by:

$$\text{recognition rate}(t) = \frac{\#S_{true}}{\#C}, \quad \text{1-precision}(t) = 1 - \frac{\#S_{true}}{V} \tag{5.15}$$

Increasing the threshold t and computing both measures yields a so-called PR-curve. Assuming a perfect descriptor this curve would pass the point (0,1), i.e. a precision and recognition rate of 1. In practice, the PR-curve of the best performing descriptor is the one for which the distance to this point is minimal. If there is more than one image pair, we calculate the mean over the PR-curves for each image pair.

Fisheye Sequence

In a first test, we simulate the tracking of subsequent camera images, by recording $I = 11$ images I_i from a planar scene with a fisheye camera (back looking camera of the helmet presented in Section 2.5). Figure 5.11 depicts five images of this sequence. To assess the matching performance of our

[1]All results in this thesis correspond to OpenCV version 3.1. which was the most up to date version at the time of writing

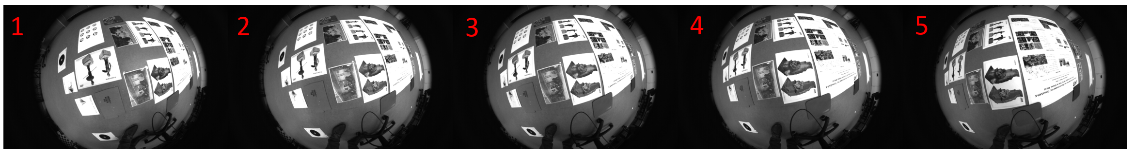


Figure 5.11. Five of the 11 fisheye images from test case 1 used for descriptor evaluation. Subsequent images are related by a homography (planar scene). The images were recorded with the back looking camera of the helmet MCS (Section 2.5).

proposed method, ground truth matches between the frames have to be determined. As the scene is planar, the mapping relating subsequent frames is given by a planar projective transformation $\mathbf{H}_{i-1,i}$ from frame $i-1$ to frame i. Thus in a first step, detected image points in the fisheye images have to be undistorted. This is achieved by using the transformation sequence from Equation 5.4-Equation 5.6. To compute ground truth homographies between subsequent images, we adopt the method used in [114]. We first align the images by manually selecting more than 10 corresponding points in each image pair yielding initial homographies $\mathbf{H}_{i-1,i}^{init}$. To refine the result, we warp image $i-1$ to image i and extract a large number of AKAZE keypoints with SURF descriptors. Note, that it is possible to use standard descriptors here, instead of fisheye optimized methods, as the baseline between frames is rather small and the images are undistorted, pre-warped and keypoints are mainly extracted close to the image center. Then, putative correspondences between subsequent images of the sequence are found by Brute-Force matching of all SURF descriptors followed by outlier removal using PROSAC [32] as implemented in OpenCV. This yields refined homographies $\mathbf{H}_{i-1,i}^{fine}$ and the final ground truth homographies can be computed by concatenating initial and refined homographies: $\mathbf{H}_{i-1,i}^{gt} = \mathbf{H}_{i-1,i}^{fine}\mathbf{H}_{i-1,i}^{init}$.

We test different properties of dBRIEF and mdBRIEF on this data set. First, the impact of learning the fixed test set on different data sets, as well as the descriptor dimension is evaluated. The results are depicted in Figure 5.12. We extract 200 corners using ORB on 4 octaves over all images. In this experiment, we do not learn masks, but rotate the tests with the given orientation of the ORB detector. In Figure 5.12b, it is visible that a larger descriptor dimension leads to better results. This effect, however, saturates for a descriptor with more than 256 bits and the performance for 128 bits is already better than a standard 256 bit BRIEF descriptor which uses random tests. Thus we chose this dimension for dBRIEF and mdBRIEF descriptors[1].

Finally, we benchmark dBRIEF (128bit) and mdBRIEF (128bit) against state-of-the-art binary descriptors, i.e. BRISK (512bit), MLDB (488bit), ORB (256bit) and BOLD (512bit). The results are depicted in Figure 5.13. In Figure 5.13a, we accounted for the patch orientation estimated by the detector, although the sequence contains almost no rotation about the view direction. BOLD performs best, followed by mdBRIEF. Note, however, that the mdBRIEF descriptor is four times smaller than the BOLD descriptor. The distorted descriptor dBRIEF still performs better than BRISK which also has a length of 512bit. In Figure 5.13b, we omit the patch orientation for BOLD, dBRIEF and mdBRIEF. Now, all descriptors without orientation perform better and mdBRIEF almost reaches the same performance as BOLD.

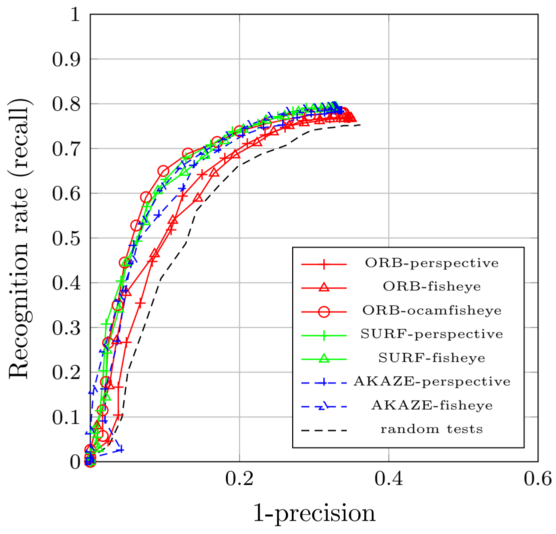
sRD-SIFT data set

The second dataset was published with the work on sRD-SIFT [114]. It consists of three image sets of 13 images respectively. Each image set was recorded with cameras having different amount of radial distortion. In addition, the scene contains significant scale and rotation change. Four randomly chosen images of every dataset are depicted in Figure 5.14.

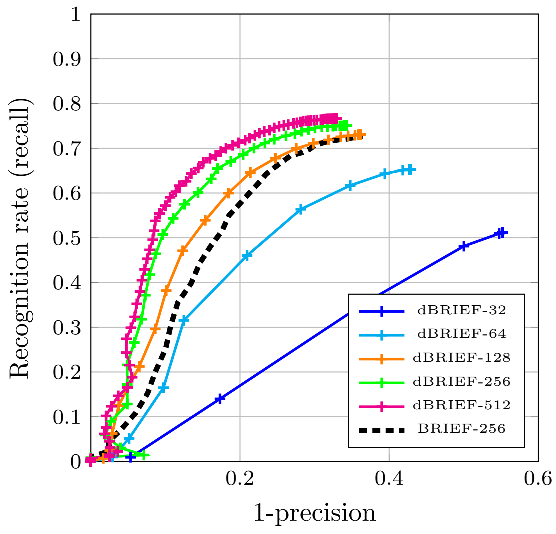
The authors use a different camera model, but attached the calibration images to the data set. Thus, we calibrate each camera using the camera model and calibration procedure introduced in Section 3 using the provided checkerboard images. The ground truth homographies are estimated according to the method used for the fisheye sequence dataset.

Again, we perform Brute Force descriptor matching and select ground truth matches with a maximal distance of three pixels. In addition, descriptors are extracted with and without patch orientation as the two of the datasets contained only little rotation about the camera view direction.

[1]However, we learned a full test set of 512 tests and thus this number is adjustable. If run-time and storage is not an important issue, performance will certainly be better with more tests.

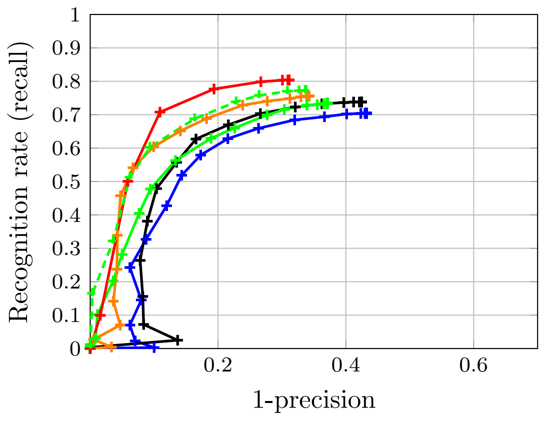


(a) Different detectors and datasets used for descriptor learning

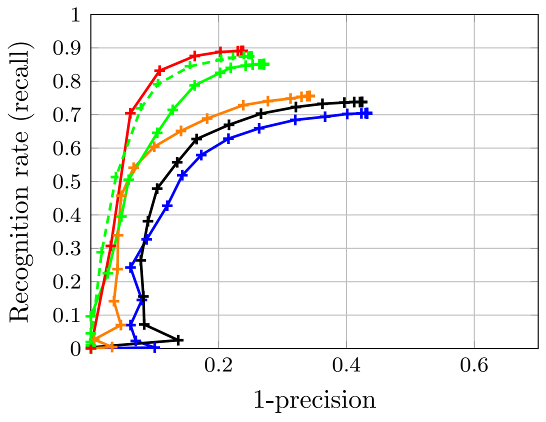


(b) Performance for different dimensions

Figure 5.12. Different investigations on offline learned test sets. (a) Depicts the descriptor performance for test sets learned on different detectors and datasets compared to each other and a random test set (BRIEF). The dimension of each test set is set to 256. *ORB-ocamfisheye* was learned on datasets from Section 2.5 and tests were distorted during the off-line learning process. (b) The descriptor dimensions is tested. Now dBRIEF uses the best performing off-line learned test set from the left figure. dBRIEF-128 already performs better than BRIEF-256 (also distorted but random tests). The performance gain from 256 to 512 bit is rather small.



(a) Including patch orientation



(b) Ignoring patch orientation for dBRIEF, mdBRIEF and BOLD

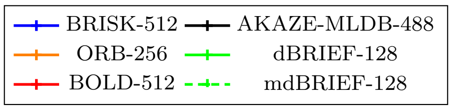


Figure 5.13. Depicted are the precision-recall curves for the fisheye sequence with 10 image pairs and almost no rotation about the camera view axis. Note, that dBRIEF and mdBRIEF use only 128 test, i.e. the descriptor has 128bits. (a) descriptor were extracted including patch orientation. (b) descriptors for BOLD, dBRIEF and mdBRIEF were extracted without rotating the descriptor pattern.

Detector	Descriptor	Detection [μs]	Description [μs]	Matching [ns]	Total [μs]
ORB	BRIEF	6	4	4	9
ORB	ORB	6	7	5	13
ORB	dBRIEF	5	18	5	23
BRISK	BRISK	16	12	9	27
ORB	mdBRIEF	5	42	8	46
ORB	BOLD	6	43	13	49
ORB	LATCH	6	59	4	65
SURF	SURF	27	41	56	69
AKAZE	AKAZE	75	56	8	131
SIFT	SIFT	108	137	119	246

Table 5.3. Extraction speed. Comparison of state-of-the-art methods. Implementations are based on OpenCV 3.1. We use the 256 bit variants of dBRIEF and mdBRIEF.

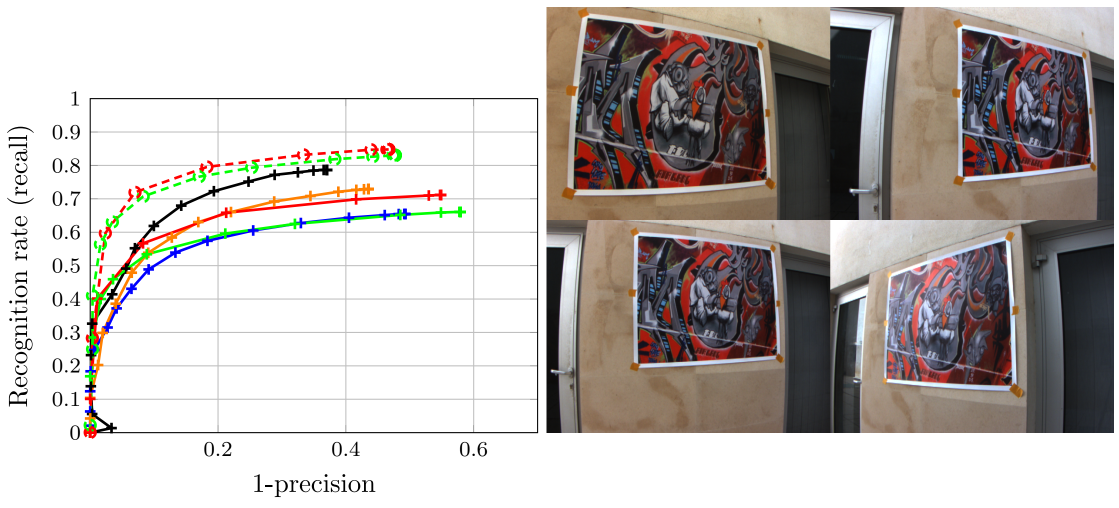
The PR-curves are depicted in Figure 5.14. Figure 5.14a shows the results for the image set with the smallest amount of distortion. Figure 5.14b shows the results for the image set with a medium amount of distortion. Figure 5.14c depicts the results for the fisheye data set. Here, we extracted a mask of the planar Graffiti region along with the ground truth homographies. The Graffiti poster occupies only a small image region and most features were detected at the floor and the doors in the right and left part of the images. Those, however, are not part of the Graffiti plane, and can hence not be transformed with the ground truth homographies and are marked as false matches.

Extraction Speed

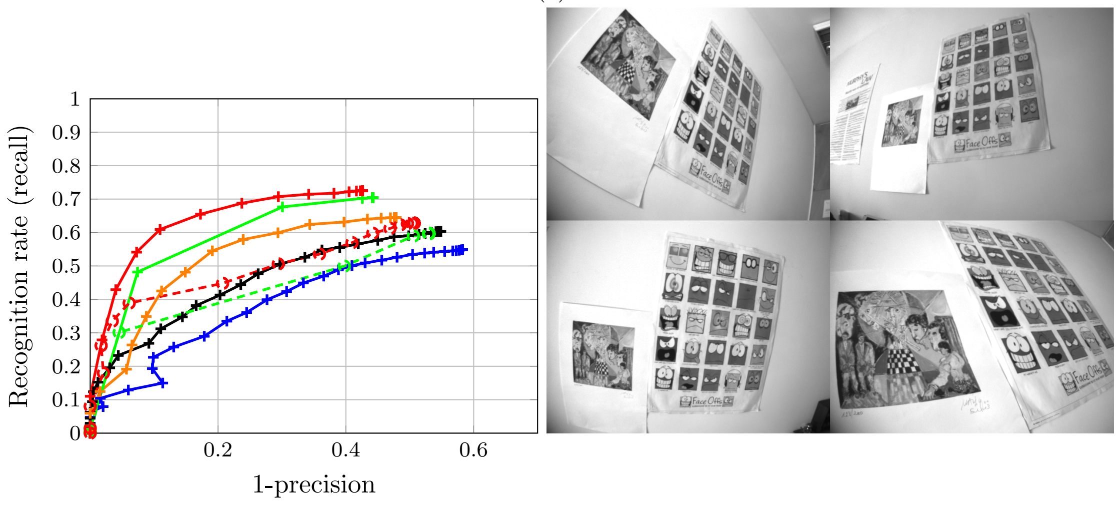
As indicated in the introduction of this chapter, the total amount of time to detect, describe and match keypoints is important to evaluate if the application of a specific detector-desciptor combination is practicable. Our two descriptor variants dBRIEF and mdBRIEF are implemented in C++ and can be used as an add-on to OpenCV. Thus, the timings are directly comparable to the OpenCV implementations. For mdBRIEF, the mask learning is performed in parallel on 4 threads. Table 5.3 shows timings for most state-of-the-art detector-descriptor combinations that are available through OpenCV. Obviously, the mask learning as well as the distortion of the test set has its price. But still, the combination of dBRIEF and mdBRIEF with the fast ORB detector leaves us with real time capable timings, apart from being more accurate than other detector-descriptor combinations as shown in the previous section. From Table 5.3 the additional effort of matching masked descriptors is visible.

5.6 Summary

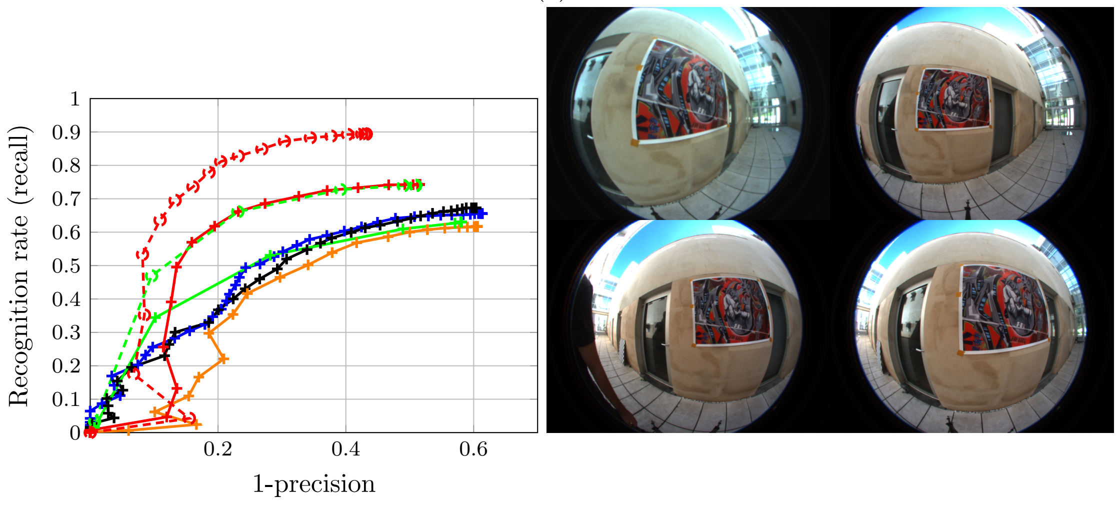
In this chapter, a modified version of the BRIEF descriptor was introduced. At the beginning, basic concepts of keypoint detection, patch description and matching were recalled. Then, the current state-of-the-art was analyzed and recapitulated. From this analysis it was possible to deduce possible detector-descriptor combinations for the task at hand. As the given methods were basically developed for perspective cameras, we combined the advantages of BRIEF, ORB and BOLD to come up with a distorted version of a binary test based descriptor. Various simulations emphasized the advantages of using the interior orientation of the camera to distort the binary test set. Finally the developed dBRIEF and mdBRIEF descriptors were tested on real data and the performance was evaluated in terms of precision and recall. Both descriptors show good results and outperform other state-of-the-art descriptors, whilst being real-time capable.



(a)



(b)



(c)

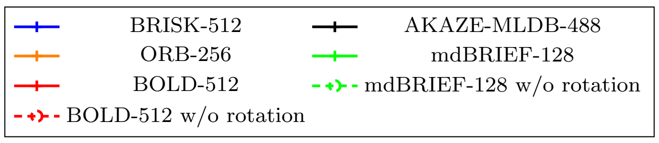


Figure 5.14. sRD-SIFT data set. For each image we extract the 300 strongest keypoints. For BRISK, ORB and AKAZE we use the standard settings. The BOLD descriptor is extracted on the same patches as the mdBRIEF descriptor.

6. Multi-Camera System Self-Localization in Textureless Building Models

In this chapter, a method for the initial image-based self-localization of the MCS w.r.t a textureless 3D building model is proposed. From this initial estimate, the continuous pose estimation can commence and the camera images can be augmented with task related information. This step is crucial, as a falsely estimated initial position within the building model will inevitably lead to incoherent information overlay and an integration, e.g. into a planning process, is hardly possible.

The general idea is that the process of estimating an initial location is supposed to run in real-time and preferably without user interaction. In addition, the self-localization is supposed to be solely based on image content. Absolute positioning sensors, such as GPS, will not be available indoors or under ground and the installation of other indoor positioning sensors or passpoints on varying construction sites would be cumbersome.

Generally, the task of computing the initial MCS pose can be narrowed down to calculating the absolute pose of the MCS w.r.t some world coordinate system. Having established three 2D-3D correspondences, i.e. three world points and their projections withing the image, would be sufficient to solve for the absolute orientation, e.g. using UPnP (cf. Chapter 4.3). The actual challenge is to robustly determine these image-to-world-points correspondences.

In this thesis, two scenarios are possible. In the first use case, the planner already visited parts of the construction site and the system stored a database of 2D features to 3D map point correspondences. Then, when the planner starts the system, features can be extracted from the live video and the database can be queried for matches. Using robust, statistical sampling methods such as RANSAC, an initial pose can be obtained. This procedure will also be used in the next chapter, to re-localize the MCS from tracking failure. The second scenario is more challenging. If the planner visits parts of the construction site that are unknown to the system or even starts the system for the first time, no visual recognition database is available. In this case, the only clue is the untextured 3D building model and the task is to find correspondences between this model and the camera images.

In the following, the state-of-the-art in this field is recapitulated. Then, we propose two methods for model-based self-localization and perform some experiments followed by a conclusion and some ideas for future work. Parts of this chapter have been published in the conference paper:

> S. Urban, J. Leitloff, S. Wursthorn, and S. Hinz. Self-Localization of a Multi-Fisheye Camera Based Augmented Reality System in Textureless 3D Building Models. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science*, 2:43–48, 2013.

6.1 State-of-the-Art

Estimating the initial pose of a camera w.r.t a 3D model of the scene is a challenging task, especially if the model is large. In general, the first step to establish 3D to 2D correspondences between model features and their projections in the camera image is to detect projected parts of the model (or the whole model itself).

In literature, detection and recognition of objects is usually divided into two reversed strategies namely bottom-up and top-down. In bottom-up like recognition schemes, features are extracted from images without imposing a-prior knowledge about the model. Subsequently, these features are clustered somehow (often by machine learning methods) and a model is composed. In the case of 3D building models, the features are usually also used to perform sparse or dense reconstruction using large sets of images [3]. In contrast, top-down like recognition schemes employ knowledge about an a-priori given model (e.g. 3D mesh) and predict (often by view sampling and rendering), how its image representation looks like. Then, given an image of the model from a real camera, the camera pose can be derived in the model coordinate system.

Bottom-Up Recognition

Many of the methods in this category use Bag-of-Words (BoW) [80] like image representations. Here, the first step is to create a vocabulary of words (image features) from a large set of images. Then, given a new image, a BoW representation can be determined by creating a sparse histogram of words that appear in this image. Given a set of BoW representations, a recognition database can be created and queried if the camera re-visits the same place. Thus in this case, instead of establishing 3D to 2D correspondences directly, image to image matching is performed a-priori

A successful place recognition system, which is especially used for loop closing, is FAB-MAP [36]. The system learns a co-visibility probability for each word off-line. FAB-MAP relies on SURF features as visual words. In [59] the authors introduce the Bag-of-Binary-Words where binary features are used as word representations, making it applicable in real-time systems. In [160] the vocabulary tree construction is improved by analyzing the information gain of a visual feature. Thus, spatial constraints are exploited by counting the occurrence of a visual word at a specific location.

In [120] the authors localize a Micro Aerial Vehicle (MAV) w.r.t Google Street View reference images. As the invariance to viewpoint changes is limited in BoW methods, the authors propose to sample synthetic viewpoint changes off-line and warp the reference images accordingly. Then, features are extracted and clustered using Fast Library for Approximate Nearest Neighbor search (FLANN) [134]. Subsequently, image features from warped MAV images are extracted and matched to features queried from the search tree on-line.

A slightly different line of research is to find the camera pose w.r.t 3D models that are the result of large-scale SfM reconstruction [3] where each 3D point has one or more image descriptors assigned to it [7, 109, 125, 126, 157, 204, 215]. In [109] the authors propose an image-based, meter accurate "worldwide" pose estimation framework. The data set is created from geo-tagged Flickr photos of interesting landmarks all over the globe. Then, nearest neighbor search on a database of SIFT descriptors is performed. Instead of directly sampling the pose from matched 2D to 3D correspondences using RANSAC, co-occurance relationships between features are used as sampling priors followed by bi-directional matching. By introducing more a-priori knowledge about the model points (e.g. gravity direction, orientation, visibility and scale) the authors of [215] propose to relax the feature matching constrains and include more geometric clues into the localization procedure and achieve state-of-the-art performance. In [126], the authors propose a system that is able to localize a mobile device w.r.t to large, sparse 3D models in real-time. Therefore, the mobile device constructs a local map using a Keyframe-based SLAM system (cf. Chapter 7.1) and sends the Keyframes to a localization server. The server performs all costly computations and globally aligns the local map to a larger 3D model. The global pose estimate is then send back to the mobile device. Thus far, all methods use point descriptors as visual features. In [125], lines extracted from a line-based SfM pipeline are employed. This proves especially useful in low-textured environments. In [83] a CNN is trained to regress the current 6DoF camera pose with meter accuracy from a single input image. A big advantage of this approach is that the storage requirements for the weights of a CNN are a lot lower than storing a large 3D model with image point descriptors.

Top-Down Recognition

To perform pose estimation from a well-known 3D object, usually synthetic views of the given mesh are generated first. In [77], a method called LINEMOD for model-based training, detection and pose estimation for a RGB-D camera is presented. Views of small, untextured but colored 3D meshes are sampled on the view-sphere and saved as templates. Then, the edge orientations are extracted from color gradients and quantized. Finally, the templates can be matched against an input image. Using gradients extracted from rendered normal images of textureless 3D meshes, [194] show high-precision pose estimation for industrial application even under severe occlusion. Following the recent success of CNNs, [206] propose an object recognition and pose estimation framework. First, poses are sampled on the view-sphere and images are rendered. Then, compact descriptors are learned for each view using a loss function, that separates descriptors from dissimilar objects and poses in descriptor space. Thus far, all methods initialize the pose w.r.t small, workspace sized 3D objects. Using 2.5D city maps (2D building footprint + height), [8] initialize the first Keyframe of a SLAM system. First the camera orientation is estimated by aligning straight line segments. Then the translation is recovered by matching facades segmented from the image with facades present in the map.

Open Challenges

Most of the related work described in this section tackles the problem of re-localization of the camera pose w.r.t to an environment that was visited before. Latter is represented either by means of a recognition database build from BoW methods or by sparse SfM reconstruction. Recent approaches also employ deep-learning to regress the camera pose, but again training data is obtained by walking through the scene of interest. In addition, only the information of a single, perspective camera is used. In the case of top-down recognition, the 3D objects are most of the time small, workspace sized and some have texture information.

In this work, we use a multi-camera system composed of three fisheye cameras. To render synthetic fisheye images custom shaders (cf. Chapter 2.4) based on the cameras intrinsics have to be developed, as OpenGL only supports perspective and orthographic cameras out of the box. In addition, the 3D model is completely textureless and very large. Thus, discriminative features need to be extracted from synthetic views that somehow mirror structure that will be visible in the camera images. Furthermore, the pose that we try to recover is inside the building model and only fractions of the model will be visible. This is in contrast to most top-down recognition approaches, where the complete model is visible in the camera and thus synthetic views can be easily sampled on the view sphere around the object. In addition, the building model can be very large and thus it is not possible to synthesize every view from every possible camera pose.

6.2 Model-Based Self-Localization

In this section, our approaches to model-based self-localization are presented. In all experiments, the MCS presented in Chapter 2.5 with corresponding intrinsic and extrinsic calibration parameters is used.

The system is supposed to estimate its current pose within the 3D building model without user intervention. The whole process can be divided into two parts and is depicted in Figure 6.1. As the 3D model is static and available before the inspection, synthetic views of the model can be rendered off-line. Subsequently, descriptors for each rendered view can be extracted and clustered. Then, as soon as the self-localization takes place on-line, the pre-created descriptors can be loaded, queried and matched against the incoming descriptors extracted from the camera images. The matching and self-localization process is supposed to run in real-time. In the following, each step of the flow-chart depicted in Figure 6.1 is detailed.

6.2.1 MCS Pose Sampling

Sampling possible MCS poses within the building model is the first crucial step to self-localization. In related work [77, 194, 206], small objects that will be completely visible within the camera image are detected. Hence, camera poses are sampled on the view sphere looking towards the object that is located in the center of the sphere. They account for scale, by varying the radius of the view sphere. In [194], the authors also account for perspective effects if the object is not located at the image center during detection.

In our approach, we need to create synthetic views for more than one camera. In addition, the MCS moves inside the object and thus sampling camera poses pointing towards the center of the

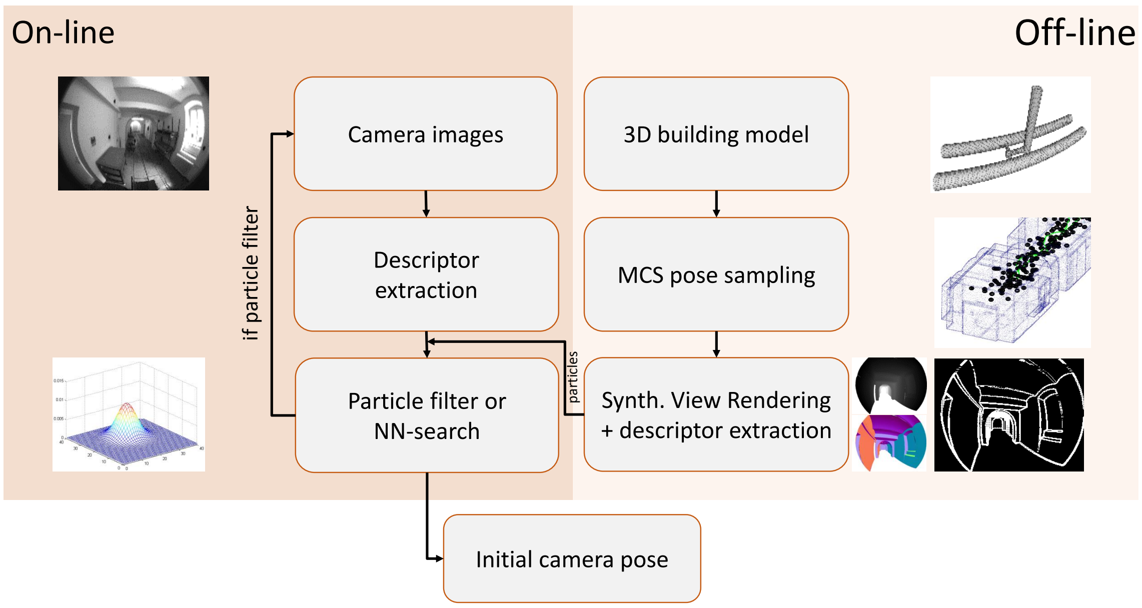


Figure 6.1. Flow-chart of the self-localization process.

object on a single view sphere will not work. At first glance, the possible MCS poses in the 3D model seem to be entirely random and thus the number of necessary pose samples would be very high. This would inevitably lead to a large search space and separation in descriptor space would become difficult. Luckily, the pose sampling can be constrained by employing some knowledge about the use case (e.g. head-mounted MCS).

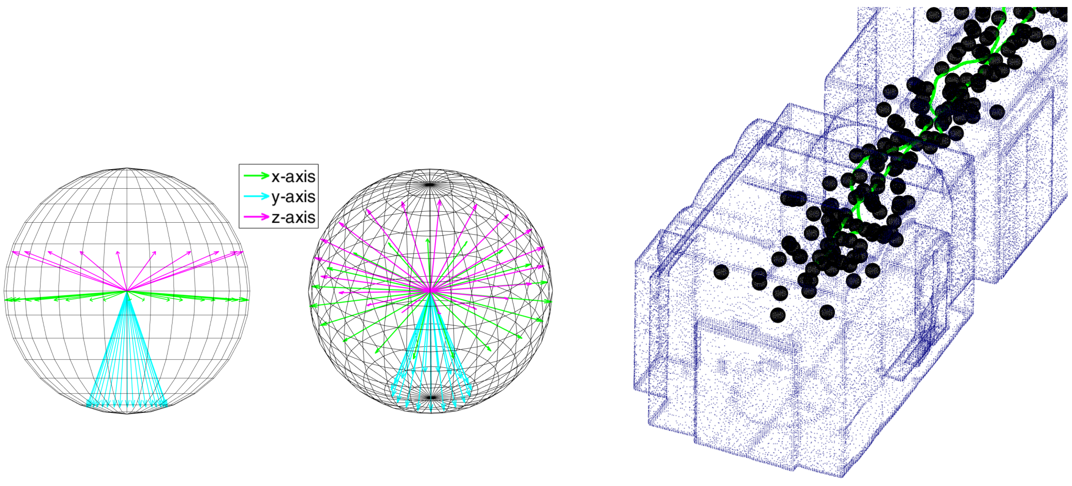
Let $\zeta_t = [\omega, x, y, z]$ be a sampled 6DoF MCS pose vector and $\mathbf{M}_t$ its homogeneous matrix representation. The parameters $\omega = [\omega_x, \omega_y, \omega_z]$ define a Rodrigues vector (cf. Section 2.2). The origin is located at position x, y, z in model coordinates and coincides with the origin of the MCS frame. The z-axis of the MCS frame is co-linear with the z-axis of the left camera and thus its viewing direction. As the initial viewing direction of the system can not be guessed in advance, we sample a rotation about the gravity direction, defined by the y-axis in this case. The rotation α about the y-axis is free from $[0, 360°]$ and is usually sampled in $10°$ steps. To rotate the MCS about the y-axis as depicted in Figure 6.2a, we can use the Rodrigues representation, as it is especially useful to describe a rotation about an axis. To this end, we rotate the system using $\omega = [0, \alpha, 0]$, that exactly represents a rotation about the y-axis with angle α. The actual view matrix $\mathbf{M}_{tc}$ for a particular camera c in the MCS is then derived using the MCS calibration matrices $\mathbf{M}_{tc} = \mathbf{M}_t\mathbf{M}_c$.

Further the operators body height is known and thus the z component can be set accordingly. This still leaves the x and y component free. To further reduce the number of pre-sampled positions, x and y are randomly sampled in a plane at level z which is depicted in Figure 6.2b. Then, we delete poses that are outside the volume as well as poses that are too close to the nearest model point, i.e. to close to the walls. The pose sampling runs completely automatic.

6.2.2 Rendering Synthetic Views

After sampling virtual MCS poses in the 3D model, synthetic images are generated. To this end, we use OpenGL with custom vertex and fragment shaders (cf. Section 2.4). Using a modern rendering pipeline automatically comes with multiple advantages:

- Speed, as the rendering is performed on the GPU and highly optimized
- Hidden-line removal
- Always appropriate level of detail
- Rendering of round objects



(a) view onto x-y plane

(b) view spheres sampled on a horizontal plane in the building model

Figure 6.2. (a) MCS view directions. The z-axis equals the viewing direction of one camera of the MCS. (b) Distribution of the view spheres on a horizontal plane in the building model. The z-coordinate of the grid corresponds approximately the height of the operators head. The ground truth trajectory (green) runs through the sampled MCS poses.

- Pixel-wise object coordinates, depth and normal information
- Seemless integration into augmented reality pipeline is possible

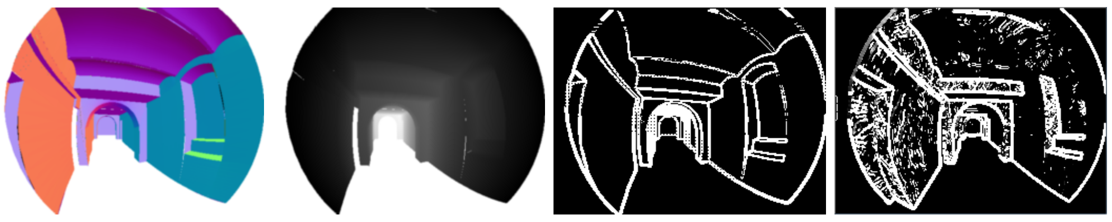
Recall that the 3D building model is textureless. Thus, extracting descriptors such as SIFT, SURF or BRIEF from rendered model images is not possible, as each relies on gray value differences within the patch around a keypoint. Rendering a textureless 3D model will not contain any gray value differences. The only visual clues that can be extracted are geometric entities. Similar to [194, 211], we render a normal and a depth image of the model and extract all visible edges. The edges contained in these views should also be visible in the real camera images. However, if information around potential edges is matched, the mapping process of the virtual camera needs to be geometrically identical with the real camera.

As the conventional rendering pipeline of OpenGL only supports perspective and orthographic cameras, we propose to use a custom vertex shader that allows us to integrate the full camera model presented in Chapter 3.2 with the back-projection function from Equation 4.16. First, the model is transformed to the particular camera reference frame using the model-view matrix, which represents the camera pose in the 3D model. Now the incident angle Θ for each vertex can be calculated and the inverse mapping (see Equation 4.12) for each viewing ray is determined. Finally the vertex is projected to the corresponding sensor position using an orthographic projection (see Equation 4.15).

After vertex processing, a fragment shader is used to color the pixels according to their normal (see Figure 6.3a) and depth (see Figure 6.3b) information respectively. At this point it is important to note, that the normal and depth values are independent of the vertex transformation. The z-value of each vertex is untouched during the mapping as well as the normal vectors. Latter are assigned to each face and vertex respectively and are calculated prior to the mapping. Otherwise the fisheye mapping would change these information and the resulting image would contain wrong normal and depth values.

6.2.3 Descriptor Extraction

Until now, two images are rendered for each camera from each virtual camera pose. In this step, the information of both images is extracted and combined yielding a holistic descriptor for a certain virtual pose. We extract the gradient information in normal and depth image by convolution with a 3×3 Sobel kernel. First, the x- and y-derivatives of both images are extracted for each channel



(a) normal image (b) depth image (c) combined edge image (d) edges of textured model

Figure 6.3. Rendered fisheye images for combined crease edge (a) and step edge (b) extraction. (c) The final edge image is a combination of edges extracted from normal and depth images. (d) Depicts the edge image extracted from rendering a textured building model with a coarse wall texture. This image is used to tune parameters for the nearest neighbor search trees (cf. Section 6.2.5).

using a kernel without Gaussian smoothing. The smoothing step can be omitted since rendered images do not contain noise. Then a gradient magnitude image $I_C = \sqrt{G_{C,y}^2 + G_{C,x}^2}$ is calculated for each channel, where $G_{C,x}$ and $G_{C,y}$ are the Sobel derivatives in x and y direction for a channel $C \in \{R, G, B, D\}$, where the first three belong to the normal and the last to the depth image.

Subsequently, each pixel in the magnitude image I_C is tested against a threshold that corresponds to the orientation change. The reason is, that the edge extraction process would detect edges at artifacts caused by surfaces and structures that are approximated by many faces, such as round structures. To avoid this, the threshold has to be set accordingly. If no edge has been detected for a particular pixel in the normal image, the depth image is tested. In this case another threshold needs to be set, which corresponds to the z-buffer range and precision. We then choose the highest magnitude $\hat{G}$ of all channels for a particular pixel u, v and extract its orientation:

$$\phi(u,v) = \arctan\left(\frac{\hat{G}_{C,y}(u,v)}{\hat{G}_{C,x}(u,v)}\right) \tag{6.1}$$

The combined gradient image is depicted in Figure 6.3c. Edges are not thinned using non maximum suppression. It showed, however, that broader edges lead to better matching results, as a perfect pixel to pixel match will barely occur and thus broader edges indirectly serve as a pull-in range for the nearest neighbor matching. Each combined edge image is then reshaped to a sparse 1D holistic feature vector $\mathbf{\Phi} = [\phi(1,1), \phi(1,2), ..., \phi(u,v)]$ that contains the orientations of all edges. In our implementation the feature extraction is carried out on images that are downscaled by a factor of 16. Here the images size is 48×30 pixels leading to a feature dimension of 1440.

6.2.4 Matching

The last step to self-localization is crucial. The off-line created, synthetic feature vectors have to be matched against the incoming camera images, using a robust distance metric. The matching is solely performed on edges. Hence, the distance metric has to be robust to occlusion, illumination change and clutter. We use the metric defined in [77] which is adapted from [172]. Latter shows its robustness to occlusions and illumination changes by taking the absolute value of the cosine of the orientation differences between search and template image edges. Dissimilar edges contribute little to the sum and negative values, caused by different illuminations, are mapped to the positive range. The distance metric is based on gradient orientations to determine a measure of similarity. Thus the distance between a search image s (the real image) and a template image t (the synthetic image) for a camera c is given by:

$$d^c(s,t) = \sum_{i=1}^{w \cdot h} |\cos\left(\mathbf{\Phi}_s^c(i) - \mathbf{\Phi}_t^c(i)\right)| \tag{6.2}$$

where $\mathbf{\Phi}_s^c$ and $\mathbf{\Phi}_{t_i}^c$ are the holistic gradient orientation descriptors of search and template image of camera c. The subscript t for the template descriptors indicates a particular pose $t = 1, .., T$ from which it was rendered in the feature creation step. The descriptor dimension is given by $w \cdot h$, i.e. is is the product of width w and height h of the search image.

Now, that the basic steps from feature creation, extraction and matching are detailed, three methods are presented, to determine the nearest virtual MCS pose to a given stream of real camera images.

6.2.5 Nearest Neighbor Initialization

The first method is to perform nearest neighbor matching between orientation descriptors. Hence, given a downsampled real image, we first extract a holistic orientation descriptor $\mathbf{\Phi}_s^c$, again using a Sobel kernel and Equation 6.1 (this time with Gaussian smoothing). The closest match from all synthetic descriptors $\mathbf{\Phi}_t^c$ can then be found by evaluating Equation 6.2 for each descriptor pair and taking the one with the largest sum. In the following, this brute-force method is called linear search.

In addition, and instead of matching each descriptor $\mathbf{\Phi}_t^c$ from all pre-sampled poses to the current image $\mathbf{\Phi}_s^c$, we use FLANN [134] to perform fast approximate nearest neighbor search. We experimented with multiple randomized kd-trees and hierarchical trees that cluster the data using k-means. Both trees have their advantages and usually the choice is a trade-off between speed and accuracy. To assess various tree parameters we performed tests with simulated data that are detailed in the next section.

To exploit the information of all cameras, we also create a panoramic orientation descriptor $\mathbf{\Phi}^p = [\mathbf{\Phi}^1, \mathbf{\Phi}^2, ..., \mathbf{\Phi}^c]$ by concatenation of all descriptors of each camera. Again, linear or approximate nearest neighbor matching can be performed over these extended descriptors.

Clearly, the biggest advantage of approximate nearest neighbor initialization is speed. The disadvantage is that the search will find a nearest neighbor for each incoming set of camera images and it is difficult to decide whether the current pose is correct or not.

Thus, we propose a second methodology that exploits the MCS configuration in a different way than concatenating multiple descriptors. Each template descriptor $\mathbf{\Phi}_t^c$ has a sampled MCS pose ζ_t attached to it from which it was rendered. In total there will be N descriptors but only N/C MCS poses if C is number of cameras, as multiple images are rendered from each MCS pose ζ_t. We can exploit this information to perform a geometric verification of the initialized pose. To this end, we match each incoming image separately, i.e. we query the database of synthetic descriptors for each image descriptor $\mathbf{\Phi}_s^c$. This yields a set S of C candidate poses $S = [\zeta_t^1, \zeta_t^2, .., \zeta_t^c]$, one for each camera. Ideally, all matches would point to the same candidate pose ζ_t, the pairwise L_2 norms between all putative pose vectors would be zero and the real MCS pose in the model is known. In practice, we observed that a perfect match rarely occurs, due to occlusions, clutter, lack of structure and nearby poses that have similar synthetic descriptors. Thus, we declare an initialization as successful if the pairwise L_2 norms of set S are below a threshold τ_t for the position and τ_α[1] for the angle part of the pose vectors. In this way, we eliminated the biggest disadvantage of nearest neighbor matching and introduce a geometric verification step. This method will be abbreviated with *multi* in the evaluation section.

6.2.6 Particle Filter Initialization

Particle filters have a long history in robotics and computer vision, where they are mainly used for localization [54, 207], navigation [65] and tracking [139] purposes. For a detailed overview, the reader is referred to [9]. In general, particle filters are sequential Monte Carlo methods that approximate an a-posteriori Probability Density Function (pdf) $p(\zeta|\mathbf{z})$ over our state space ζ given current measurements $\mathbf{z}$. The advantage of particle filters compared to other information filters such as the Kalman Filter is that they can represent multi-modal, non-Gaussian distributions.

In our special case, the particle state space is represented by all pre-sampled camera poses ζ_t with attached orientation descriptors $\mathbf{\Phi}_t$. Now, the goal is to assign each particle $x_t = [\zeta_t, \mathbf{\Phi}_t]$ a weight w_t, that corresponds to the probability of a particle to be the true state, and thus the initialized pose. The weights are normalized:

$$\hat{w}_t = \frac{w_t}{\sum_{t=1}^{N} w_t} \tag{6.3}$$

[1]As we use the Rodrigues rotation representation the L_2 the norm of the vector difference directly corresponds to the angle of rotation in radians and a threshold in radians is feasible.

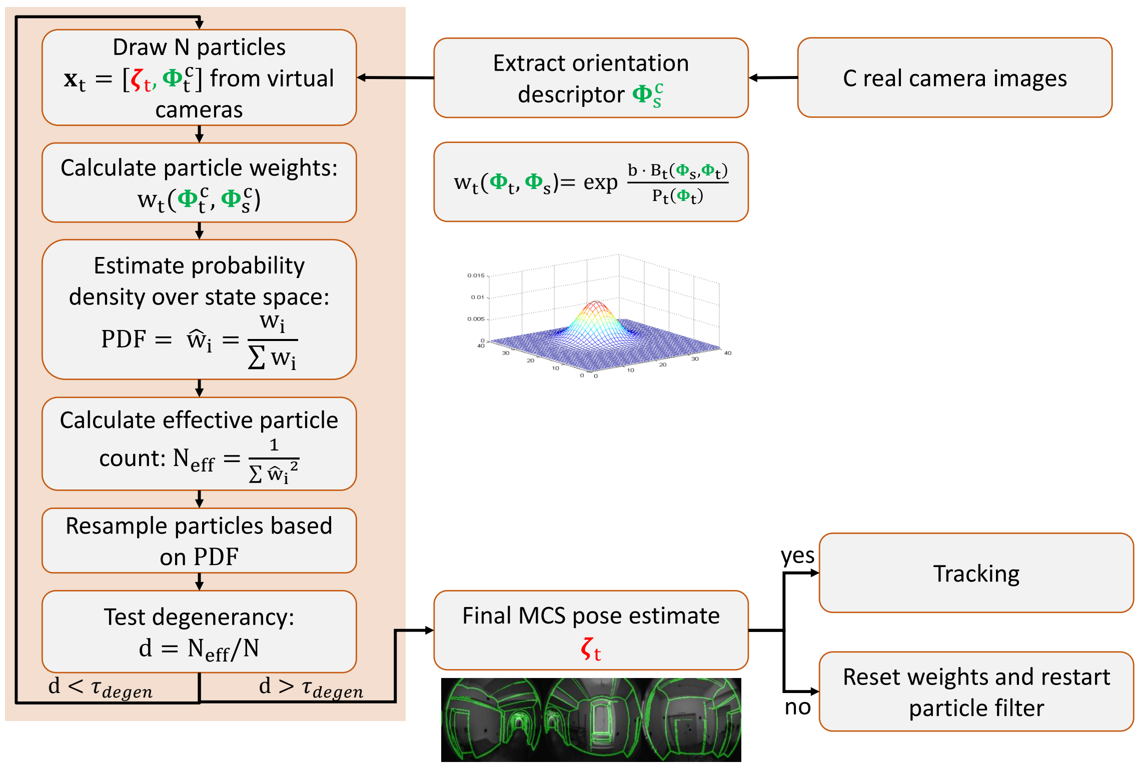

Draw N particles $\mathbf{x}_t = [\zeta_t, \Phi_t^c]$ from virtual cameras
Extract orientation descriptor Φ_s^c
C real camera images
Calculate particle weights: $w_t(\Phi_t^c, \Phi_s^c)$
$w_t(\Phi_t, \Phi_s) = \exp \frac{b \cdot B_t(\Phi_s, \Phi_t)}{P_t(\Phi_t)}$
Estimate probability density over state space: $\text{PDF} = \hat{w}_i = \frac{w_i}{\sum w_i}$
Calculate effective particle count: $N_{eff} = \frac{1}{\sum \hat{w}_i^2}$
Resample particles based on PDF
Test degenerancy: $d = N_{eff}/N$
$d < \tau_{degen}$
$d > \tau_{degen}$
Final MCS pose estimate ζ_t
yes
Tracking
no
Reset weights and restart particle filter

Figure 6.4. Flow-chart of the particle filter self-localization process.

and sum to one: $\sum \hat{w}_t = 1$.

Instead of instantly taking the state with the biggest weight for the first set of images (as done in nearest neighbor matching), the filter recursively estimates the particle weights with each incoming set of images until it degenerates into a few particles, that have a high probability of corresponding to the true pose. To exclude particles with low weights from the state space over time, sample importance re-sampling (SIR) is performed (see [9]).

A flow chart of our particle filter algorithm is depicted in Figure 6.4 and is described in the following:

(1) The first step is to extract the orientation descriptors $\mathbf{\Phi}_s^c$ for each image of the real MCS.

(2) Then, N particles $\mathbf{x}_t = [\zeta_t, \mathbf{\Phi}_t^c]$ are randomly drawn from the total set of available particles (virtual poses). We draw particles with replacement, i.e. some will occur multiple times. The probability of a particle to be drawn is proportional to its weight, and thus the probability of the particular particle. Particles that have a higher likelihood of corresponding to the true MCS pose will survive longer. We use the domain independent, multinomial importance re-sampling algorithm that was presented in [9]. The pseudo-code is given in Algorithm 1.

(3) The next step is to recalculate the weights for each particle. Therefore we match the descriptors $\mathbf{\Phi}_s^c$ to the current set of particles $\mathbf{x}_t$. Again the distance metric defined in Equation 6.2 is used, but with some modifications. We observed, that the distances obtained by Equation 6.2 do not differ enough in descriptor space to serve as meaningful weights. The descriptor differences are small and the re-sampling step will not yield particles with higher probability. Thus, we use the following equation to calculate the particle weights, which is similar to the edge tracker in [89]:

$$w_t(\mathbf{\Phi}_s, \mathbf{\Phi}_t) = \mathrm{e}^{b \frac{B_t(\mathbf{\Phi}_s, \mathbf{\Phi}_t)}{P_t(\mathbf{\Phi}_t)}} \tag{6.4}$$

where b is a scale factor that increases the relative weight of stronger particles and P is the number of non-zero entries in the template descriptor $\mathbf{\Phi}_t$. It normalizes out the number of tested descriptor values. Otherwise, the particle filter would be drawn to descriptor with more non-zero entries. The parameter B is calculated by evaluating Equation 6.2 between search and template descriptor and counting the entries for which the orientation difference is below some threshold τ_B, usually set to $10°$.

(4) Having calculated weights for each particle using Equation 6.4, the pdf can now be estimated by normalizing the particle weights using Equation 6.3. As particles with higher weight get drawn more often during re-sampling, the effective number of remaining particles can be obtained by $N_{eff}^{-1} = \sum_{t=1}^{N}(\hat{w}_t^2)$ (see [9]) which actually reflects the variance that remains in the weights.

(5) Finally, we test the degeneracy of the particle filter, by calculating the ratio between N_{eff} and the original number of particles N. If this ratio exceeds some threshold τ_{degen}, i.e. most particles accumulated into a few remaining putative poses with high likelihood, we stop the filter. Then the final pose is taken as the particle that occurs the most in the remaining set of particles.

The particle filter tends to delay the initialization until it is sure about its current state. Yet there exists no geometric verification of the initialized pose. The particle filter can both be used with the panoramic descriptors $\mathbf{\Phi}^p$ as well as separately for each camera $\mathbf{\Phi}^c$.

Algorithm 1 Multinomial re-sampling

```
Require: normalized weights $\hat{w}_t$, number of particles $N$, particles $x$
 1: function RESAMPLE($\hat{w}_i$,$x_i$)
 2:     $c_1 = 0$                                   ▷ initialize cumulative sum
 3:     for $i = 2$ to $N$ do
 4:         $c_i = c_{i-1} + \hat{w}_i$
 5:     end for
 6:     for $k = 1$ to $N$ do
 7:         sample = $\mathcal{U}[0.0, 1.0]$         ▷ with $\mathcal{U}$ being the uniform distribution
 8:         $j = 0$
 9:         while $c_j < sample$ do
10:             $j = j + 1$
11:         end while
12:         $x_k = x_j$
13:     end for
14:     return $x_k$
15: end function
```

6.3 Experiments and Results

In this section, the proposed methods for model-based self-localization of a MCS are evaluated on the lasertracker ground-truth trajectory 1 (see Section 2.5). First, simulations will be performed to find and adjust parameters especially for the approximate nearest neighbor matching. Therefore, a coarse wall texture is attached to the model to simulate missing edges and image noise. An example of the edge image extracted from the rendered textured model is depicted in Figure 6.3d. For all experiments, the 3D mesh shown in Figure 6.5 is used. Again, like in Chapter 5.15, the performance is measured in terms of recognition rate (or recall), i.e. the number of correctly detected poses divided by the total number of poses. For all methods that do not initialize each MCS pose (particle filter and *multi*), the total number of ground truth poses is reduced to the number of initializations.

6.3.1 Simulations

First, the linear search is applied to find the corresponding descriptor vector from all pre-rendered poses, as it is the most accurate method. For the linear search, each descriptor $\mathbf{\Phi}_s$ from the simulated real image, (the one having a coarse wall texture to simulate noise, see Figure 6.3d) is matched to all descriptors $\mathbf{\Phi}_t$ from the pre-sampled poses. We use only one camera for the simulations. The pose ζ_t is sampled on a regular x-y grid. The distance between neighboring poses is 0.5m and the angular increment for the rotation about the y-axis is set to $10°$ for the simulations.

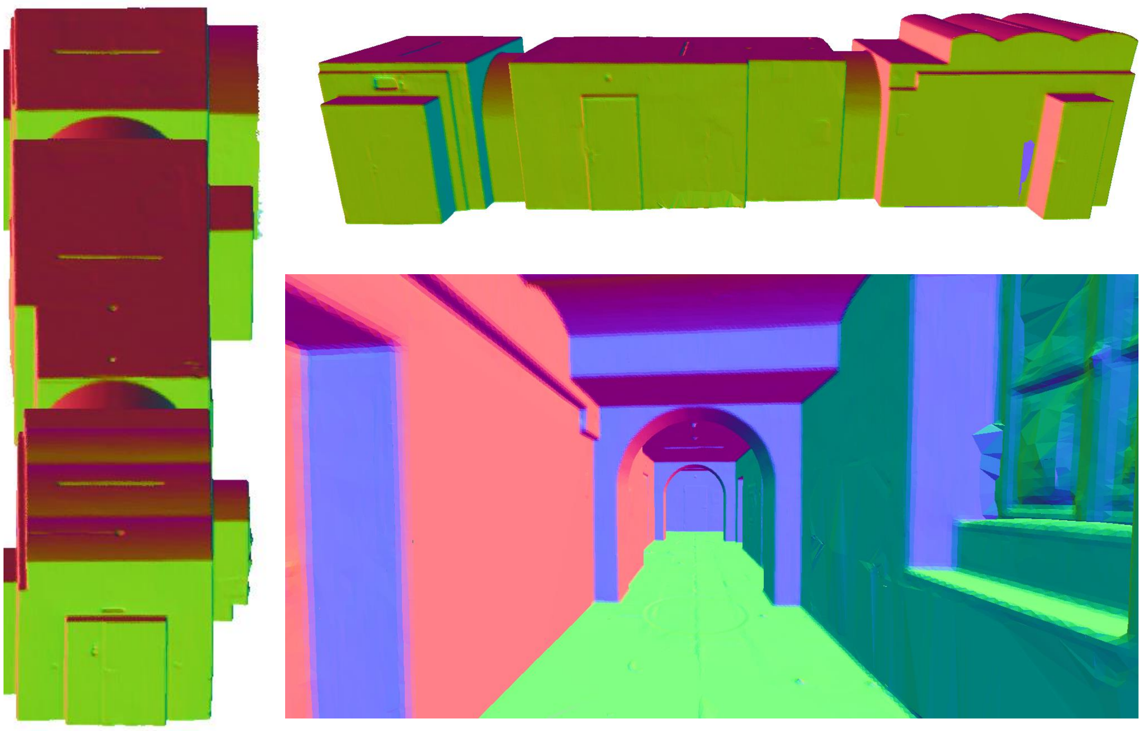


Figure 6.5. We use a 3D building model, that was meshed from a laserscanner point cloud. Depicted are different views of the model rendered using a fragment shader that displays vertex normal directions.

We then add an increasing Gaussian noise to position and angle values of ζ_s of the simulated image respectively. The noise is added separately to position and angle to test the robustness of the method against both parameters. The result is depicted in Figure 6.6. One can see that, for increasing angle noise, the precision reaches 50% when the noise reaches 5°. This corresponds exactly to half of the sampled angle increment. For noisy positions, the recognition rate remains good up to a noise of 0.15m. This is one third of the sampling density on the grid. Next we use FLANN to cluster the template descriptors ζ_t and create search trees for fast approximate nearest neighbor search. For the randomized kd-trees basically only the number of trees influences build time, query speed and recall. For hierarchical k-means clustered trees, the number of clustering iterations as well as the branching factor has to be tuned. Especially the tree build time will increase significantly if the clustering is iterated to often.

Again, we use the set of template descriptors $\mathbf{\Phi}_t$ to build different trees. In total we sampled 9000 descriptors for the simulation experiments. Figure 6.7 depicts results for different aspects of

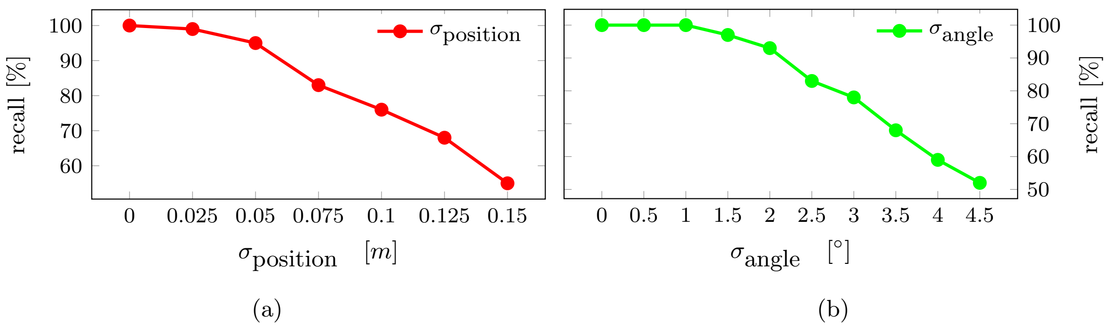
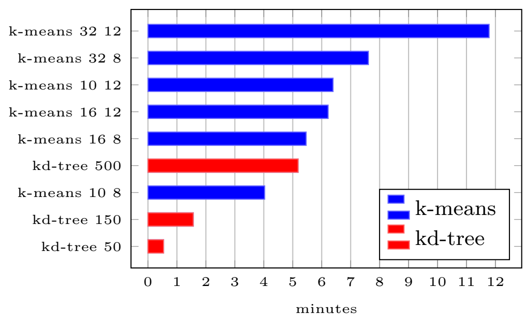
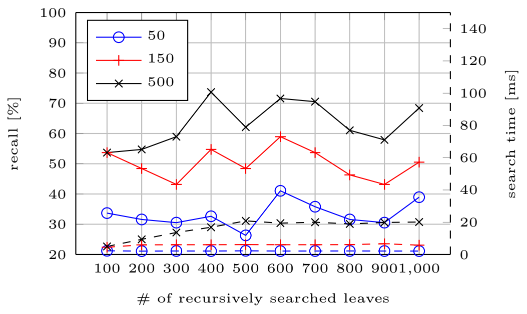


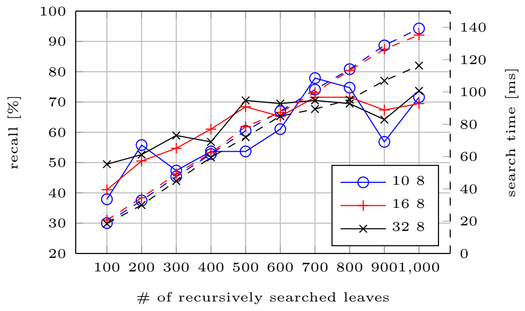
Figure 6.6. Recognition rate of matching simulated real and synthetic images with increasing noise to the ground truth (a) position (b) orientation.



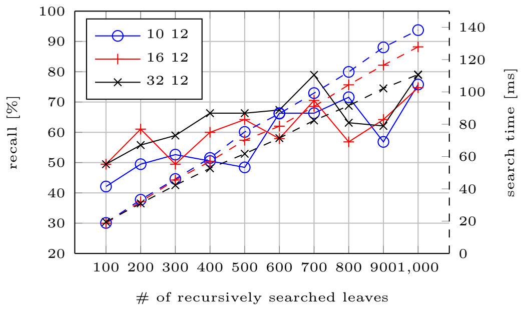
(a) comparison of build times for 9000 features in minutes



(b) kd-tree with randomized trees



(c) hierachical trees with k-means clustering



(d) hierachical trees with k-means clustering

Figure 6.7. Search precision vs. speed (dashed lines) for 9 different trees (number of features for all plots: 9000). Angle noise was set to $\sigma_{\text{angle}} = 1.5°$, position noise to $\sigma_{\text{pos}} = 0.05$m. The precision is related to the precision at this particular noise level for the linear search. Parameters presented in the legends:(b) kd-tree *number of trees*, (c)-(d) k-means *branching factor* and *number of iterations*.

the simulation. First, we analyzed the tree build time for different parameter configurations. The results are depicted in Figure 6.7a. In general, the build time plays a minor role as it can be done off-line. Clearly, the build time increases significantly for higher tree counts (kd-tree) and a higher number of branching and clustering iterations (k-means).

Next, we analyzed the search time and recognition rate of different tree parameter configurations. We set the position and noise level for the query pose to 0.05m and 1.5°, respectively, where the linear search still reaches more than 95% recognition rate. In Figure 6.7b, the results for randomized kd-trees are depicted. Clearly, the recognition rate is highly correlated with the number of trees. The more trees, the higher the recall. The influence of the number of recursive leave searches, however, shows no obvious trend. This number indicates how many leaves are visited to return the approximate nearest neighbor. The search time is very low, even for a large number of trees and would run at frame rate.

Figure 6.7c and Figure 6.7d depict the results for the hierarchical trees with k-means clustering. In contrast to the kd-trees, the number of searched leaves shows a slight trend. The more iterations are performed the better the recall. The search time, however, also increases significantly and approaches the time that is needed to perform the linear search over all descriptors. The different cluster iterations between Figure 6.7c and Figure 6.7d as well as the branching factors tend to have a negligible impact, and thus the configuration with a lower build time can be used. In the real world experiments, we thus use a branching factor of 10 with 8 clustering iterations and search 700 leaves. For the kd-tree we use 500 trees and search 400 leaves.

6.3.2 Real World Ground Truth

To assess the developed methods for model-based self-localization in a real world scenario, the MCS from Chapter 2.5 is used. The synthetic images are all rendered with the actual camera parameters of the corresponding camera in the MCS. All real images are resized with a scale factor of 16 and the synthetic images are directly rendered to an appropriate image size of 48×30 pixels.

Thus, the descriptor dimension for $\mathbf{\Phi}_t^c$ and $\mathbf{\Phi}_s^c$ is 1440 and 4320 for the panoramic descriptors $\mathbf{\Phi}_t^p$ and $\mathbf{\Phi}_s^p$ respectively.

We assume that the operator stands upright. The inclination of the y-axis of the calibrated MCS w.r.t the gravity direction is taken from various experiments and is set to 10°. Then, the x and y coordinates of ζ_t are randomly sampled and positions too close to the walls and outside of the model are removed. Subsequently, the MCS orientation around the gravity direction is sampled in $\pi/10$ steps.

We use the first ground truth trajectory detailed in Chapter 2.5. The 3D model is transformed into the lasertracker coordinate system using Horn's closed-form solution [78] over 14 identical points. The trajectory contains 1019 ground truth MCS poses ζ_t^{gt}. To put the results of our generic self-localization algorithm with randomly sampled poses into perspective, we first create baseline recognition rates for each method, i.e. we assess the quality of the localization from the actual ground-truth poses. If the self-localization worked perfectly, all methods would always return the exact ground truth pose ζ_t^{gt} from which the template descriptor was rendered. As the real images contain a lot of occlusions, challenging lightning conditions and moving objects, this will certainly not be the case. In addition, the descriptors are created from a high pyramid level, i.e. the initialization is invariant to scale changes to some degree and thus the methods can not distinguish between sampled poses that are very close together. The self-localization over multiple-scales is subject to future work.

Baseline

We first render the set of template descriptors $\mathbf{\Phi}_t$ for all ground truth poses ζ_t^{gt} and evaluate each method creating a baseline recognition rate for each method. These baseline results indicate how well the matching between rendered and real images based on edge orientation works.

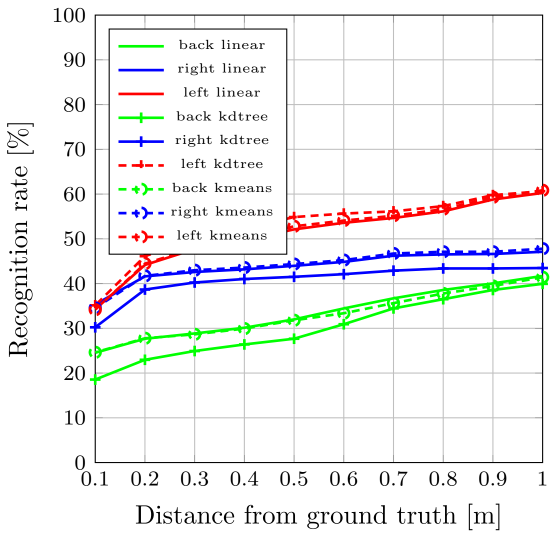
We calculate the recognition rate of each method for multiple distances to the ground truth pose due to the high pyramid level at which the descriptor is created. The recognition rate is calculated for 10 distance intervals and a localization is only accepted if the orientation angle between the viewing direction (z-axis) is equal or smaller than the sampling interval $\pi/10$ at any distance interval.

The results for the baseline recognition rates is depicted in Figure 6.8. In Figure 6.8a the (approximate) nearest neighbor initialization is depicted. Each camera was initialized separately using linear as well as fast approximate nearest neighbor search. A pose was estimated for each frame of the trajectory as this methods simply queries all available descriptors $\mathbf{\Phi}_t^c$. Obviously the initialization does not reach the same recognition rates over the trajectory for different cameras and hence different views of the model. From this result, we can conclude, that it should be beneficial to combine the pose estimates or the features of all cameras.

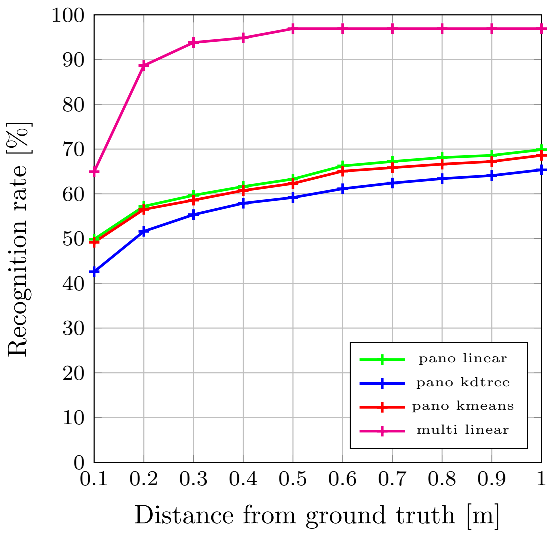
This is depicted in Figure 6.8b. Here, we perform nearest neighbor search over the panoramic descriptors $\mathbf{\Phi}_t^p$ as well as the *multi* method. Again the panoramic descriptor method initialized every set of frames coming as a stream from the MCS. The *multi* method, however, performs a geometric verification step as explained in Section 6.2.5. Hence, less poses are initialized, but obviously the recognition rate is a lot higher and reaches 97% for a distance to the ground truth of 0.5m. Below in Figure 6.8c and Figure 6.8d the initialization recognition rates for the particle filter are depicted. In Figure 6.8c each camera was initialized separately and in Figure 6.8d the panoramic descriptors were used. Especially the recognition rate for smaller distance errors is slightly higher for the particle filter. In addition, the performance for camera "back" is better

Random Poses

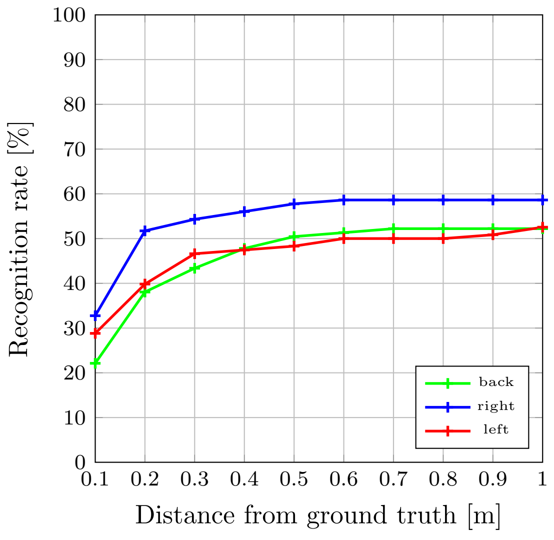
Finally, we sample 1000 poses on the x-y grid and remove the ones that lie outside the building model, as well as poses that are too close to the walls. Then, for each of the 1000 poses the virtual camera system is rotated according to the pose sampling procedure from Section 6.2.1 and we end up with 18600 MCS poses. Subsequently, we create FLANN indexes for this set of features, and execute the self-localizations methods. The results are depicted in Figure 6.9. For single camera initialization the particle filter (Figure 6.8c performs slightly better than initializing each pose of the trajectory using FLANN (Figure 6.8a). The panoramic particle filter does not perform better than the single camera particle filter. The *multi* method again outperforms all other methods by a large margin and reaches recognition rates of over 80%. It initializes only 47 poses out of 1019 but with a high recall.



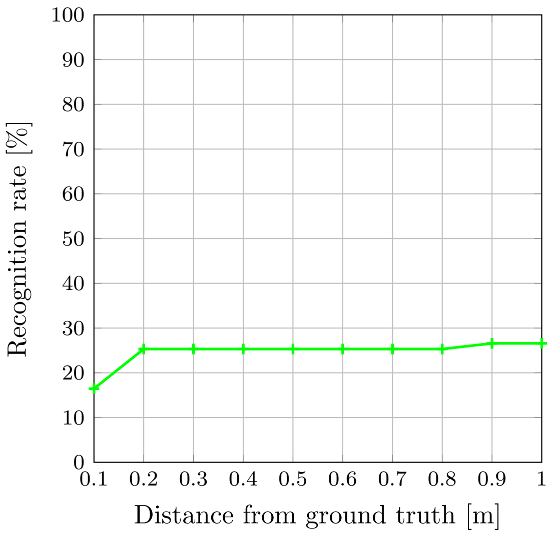
(a) Baseline recall curves for each camera and all three matching methods.



(b) Baseline recall curves for panoramic feature localization.

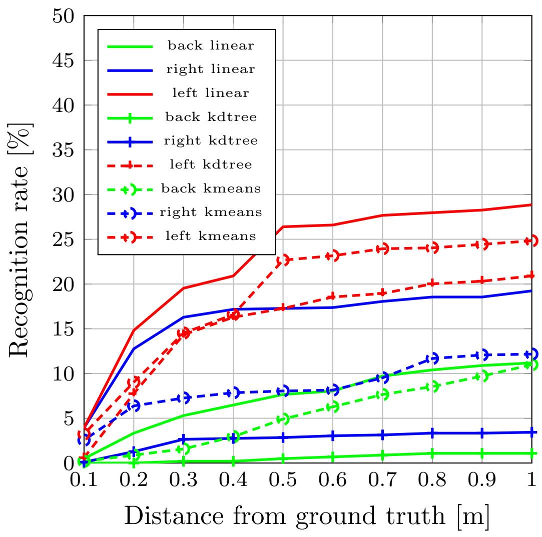


(c) Baseline recall curves for the particle filter for each camera.

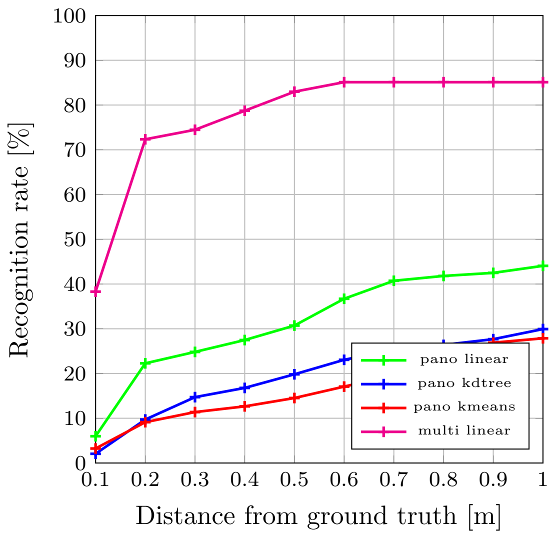


(d) Baseline recall curves for the particle filter using panoramic feature vectors.

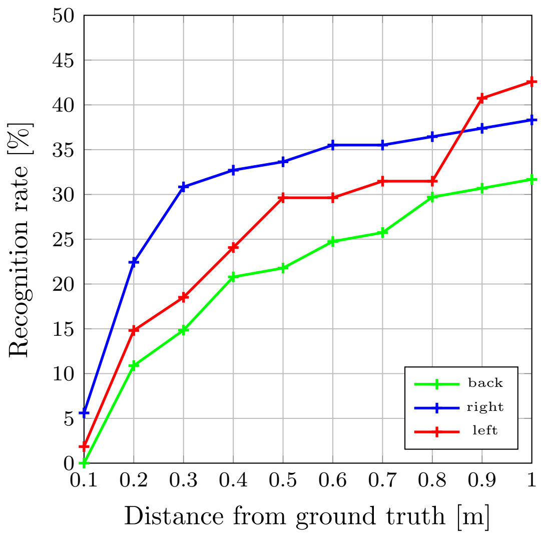
Figure 6.8. Depicted are the baseline recognition rates. Here, we used the ground-truth poses to render the template descriptors Φ_t. This experiment is conducted to explore the maximum possible performance of the methods on this dataset. Cameras are: back, right, left according to the helmet camera system Section 2.5 and the nearest neighbor methods are: linear, kdtree and kmeans.



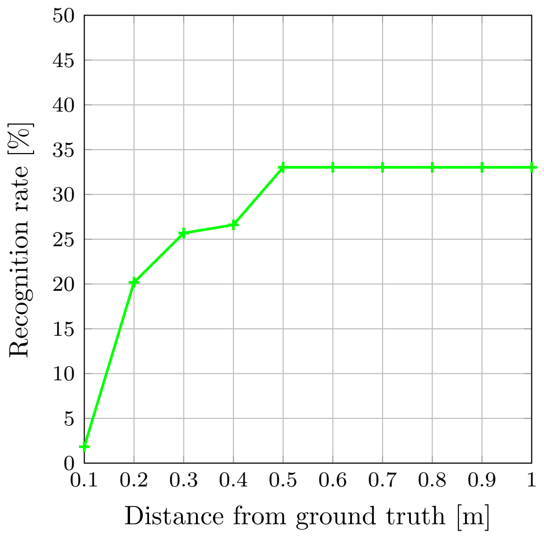
(a) Baseline recall curves for each camera and all three matching methods.



(b) Baseline recall curves for panoramic feature localization.



(c) Baseline recall curves for panoramic feature localization.



(d) Baseline recall curves for particle filter panoramic feature localization.

Figure 6.9. Depicted are the recognition rates for 18600 randomly sampled poses for which template descriptors Φ_t are extracted. Again cameras are: back, right, left according to the helmet camera system Section 2.5 and the nearest neighbor methods are: linear, kdtree and kmeans.

6.4 Summary

In this chapter, a method for self-localization of a multi-fisheye camera system in texture-less 3D building models was presented. First, model-based initialization was motivated in the context of this work. Then, the sampling of MCS poses from which synthetic but geometrically correct images are rendered was introduced. Rendering normal and depth images of untextured models allowed us to extract descriptors based on edges that are supposed to be visible in in both the real and the synthetic images. Subsequently, two initialization methods were proposed. One is based on approximate nearest neighbor matching, the other on particle filtering. Finally, we performed simulations to tune parameters followed by experiments and evaluations of the proposed methods on a real-world dataset.

7. MultiCol-SLAM - A Multi-Fisheye SLAM System

The final chapter of this thesis combines all previous work (Chapter 3-Chapter 6) into a 3D model-supported tracking and mapping framework for multi-fisheye camera systems. The basis for a large-scale augmented reality application that is able to support planners and construction engineers on site is a robust, continuous estimation of the current position and orientation of the camera system w.r.t the observed environment (scene). In this work, latter is subject to different states in the real world (e.g. during construction) and can change over time. At the same time, the currently observed objects and feature points in the scene can have a different state. Some might be already reconstructed from previous exploration, some can be part of the underlying 3D model and are not supposed to change over time and some might be beneficial for estimating the current pose, but are likely to vanish or change in long or short-term operations. In praxis, the system will have to cope with all challenges and dynamically switch between different states.

Imagine an ongoing construction of a subway track. This application and usage scenario is depicted schematically in Figure 7.1. The planner starts the augmented reality system from an already finished part of the structure (left part of Figure 7.1) and initialized the pose from the given 3D model (see Chapter 6). Here the system can use the model as well as natural, real world features that are classified as being fixed over time, to estimate its current pose. No reconstruction has to be carried out at this time. Then, the planner walks into the tunnel and passes parts of the construction site that are partly finished or under temporary construction. Here, the system might find scene features that are already part of the model and many features that are not. The camera pose tracking can be supported by the 3D-model but mostly features are reconstructed on temporary scene elements. Approaching the current construction area, the difference between reality and planned 3D-model will be large. Thus, the camera system switches to full exploration and performs SLAM in the unknown scene.

In the remainder of this chapter, we combine strategies and related work from different fields such as computer vision and graphics, robotics and photogrammetry and develop new methods to tackle the described challenges. First, related work is recapitulated and categorized into different research directions. Then, the entire framework is described in detail. Subsequently, we perform various tests and experiments to analyze the different building blocks of our method and compare the system to related state-of-the-art methods.

7.1 SLAM - Principles and State-of-the-Art

The accurate reconstruction of the observed scene from sets of ordered images has a long history in aerial [97] and close-range photogrammetry [116]. Usually, the object and reconstruction setup is well defined and the scene observations using high-resolution cameras are well planed. Thus, the connectivity between multiple camera positions is known or easily established and off-line

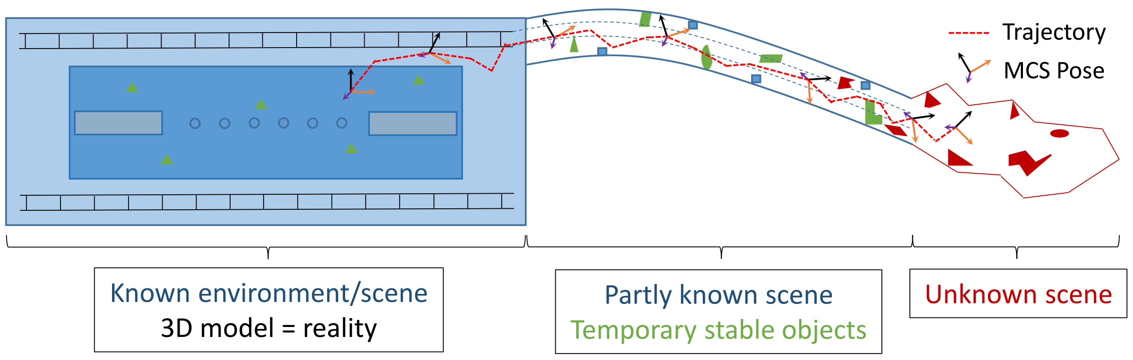


Figure 7.1. Depicted is a usage example of the proposed MCS egomotion estimation pipeline and the different states it has to handle. The planner starts in a known environment. Here the system can use the 3D model to estimate its pose and no reconstruction is necessary. Then as the planner walks through the tunnel, some parts of the structure are already finished and some can be reconstructed and classified as temporary stable. Finally, as the planner walks into unknown parts of the scene, full reconstruction and pose estimation is carried out.

bundle adjustment over the cameras and scene structure is performed yielding accurate results. In addition, initial values for the exterior and interior camera orientations are mostly available from external sensors, passpoints and accurate calibration.

In the computer vision community, the direction of research is called SfM and relaxes many constrains about scene and camera geometry that are assumed in classical photogrammetry. Advances in projective geometry [73] and visual feature research (see Chapter 5.1) enabled the off-line reconstruction of large scenes from unordered sets of images and photo collections ([3, 171, 185, 186, 189, 209, 210]). But essentially, the SfM methods solve the exact same problem as in aerial and (close-range) photogrammetry, i.e. reconstruction of scene and cameras. The main difference lies in the initialization of the bundle adjustment through direct relative orientation methods for calibrated [73, 96, 173] or uncalibrated [15, 99] cameras and a simultaneous connectivity estimation using only natural image features.

In both communities, the scene is basically assumed to be static and the reconstruction is done off-line doing batch optimization over the entire scene. Hence, correspondence information (features) can be exhaustively extracted and matched a-priori and has no temporal coherence. Latter however, is the case for live video frames from a moving camera. In addition, the pose change (baseline and orientation) between subsequent frames can be very small.

Thus, the robotics community, where sensors (including cameras) are mounted on top of moving platforms, developed filter-based methods [39, 40, 129, 130] to estimate the sensor pose from continuous sensor updates (live video). In recent years, basically two streams emerged from this line of research, namely filter- and keyframe-based SLAM techniques that will be described in more detail in the following. The basic idea behind both approaches is that not all poses, observations and uncertainties from a continuous video stream can (and have to) be integrated into the solution of the SLAM problem. Outstanding work of [41] and [178] give a detailed analysis of both methods and close the gap between SLAM, SfM and classical bundle adjustment by generalization of the entire reconstruction problem using graphical models.

Figure 7.2 depicts the structure in an undirected graph (Markov Random Field) for a toy example. In total, four frames were recorded that observe a scene consisting of five map points. The poses are connected to map points by edges (observations) and the poses themselves are connected by state transitions (e.g. inertial sensor). An optimal Bundle Adjustment (BA) solution is given (learned) by estimating the Maximum Likelihood (ML) solution for the depicted graph and could be computed efficiently in this example. Speaking in terms of graphical models, the ML solution corresponds to the joined probability distribution over all parameters (poses and map points). Now, lets grow he graph by adding poses and map points. With each frame, the computational complexity increases and quickly exceeds the runtime constrains for real-time applications. Thus, we need a way of thinning out the graph. Using a filter-based approach, historic poses are marginalized out of the joint distribution by integration over all other parameters. The resulting graph is depicted

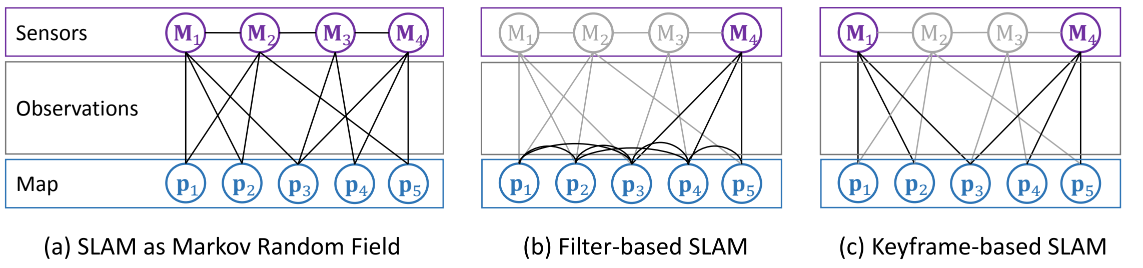


Figure 7.2. (a) SLAM as a Markov random field (based on [178]). (b) The graph is sparsified by marginalization of past poses except the current one from the graph. (c) Only keyframes are selected and all other poses, observations and object points, that are not visible in any retained keyframes are removed.

in Figure 7.2b. The issue with filter-based SLAM becomes visible. To marginalize the current pose from the joint distribution, all potentials between connected variables need to be stored and updated with every frame and thus the number of map points will be very limited.

But especially a large map and many observed features lead to accurate and robust SLAM pipelines [178]. Thus, the idea behind keyframe-based SLAM is to sparsify the graph by simply removing poses, map points and observations from the graph instead of marginalizing them from the probability distribution. In this way, classical BA can be performed which is efficient and fast to compute on sparse matrices [102, 189]

Thus in this thesis, a keyframe-based SLAM pipeline will be used as the basis for the continuous estimation of map and MCS pose. An additional benefit of using keyframes is, that the map points and observations that are connected to certain keyframes can be easily updated when the MCS revisits a place that changed.

In the following, we detail some related work. For a more detailed overview of available methods and features the reader is referred to [61, 62].

7.1.1 Keyframe-Based Approaches

In recent years, plenty of keyframe-based SLAM systems were proposed and many of them are publicly available. Probably the first real-time SLAM system that was based on performing local and global BA over keyframes is Parallel Tracking and Mapping (PTAM), by Klein and Murray[86]. One key to success was to split the tracking and mapping components of the system into separate threads running in parallel on a dual-core CPU. In this way, the mapping thread decides which camera poses to keep and store as keyframes and performs local and global BA asynchronously to the tracking thread. Latter runs at frame rate and performs matching of map point and camera pose estimation. Subsequently, PTAM was improved by adding edge features [87] and, with the increase in computing power, was implemented on a mobile phone [88]. The use of image patches as features and the lack of loop closing mechanisms already suggests that large scale operation could be an issue using PTAM (see also Table 5.1) as storing, updating and indexing of image patches is costly. In addition, the global BA is performed over the entire scene, limiting its applicability to smaller workspaces.

To extend the range of PTAM, the authors of [177] propose a so-called scale-drift aware SLAM system. First, the keyframe decisions and feature initialization is moved to the tracking thread. Despite having a higher computational burden, the tracking becomes more stable, as features and keyframes are not initialized asynchronously and are immediately available for pose estimation. Due to the incremental nature of SLAM, the trajectory estimates start to drift over time, i.e. if the camera visits the same place twice, it will have a different exterior orientation and the reprojected map points exhibit large residuals to the measured image features. Still, if one is able to automatically detect similar places (place recognition [61]), loop closing can be performed. This is very similar to (closed) traverses in geodesy, where the loop misclosure error is distributed over all poses in the chain. The loop closing mechanism in [177] is based on SURF features and a dense surface model. Then, the trajectory is corrected and optimized over similarity transformations, i.e. also correcting scale-drift.

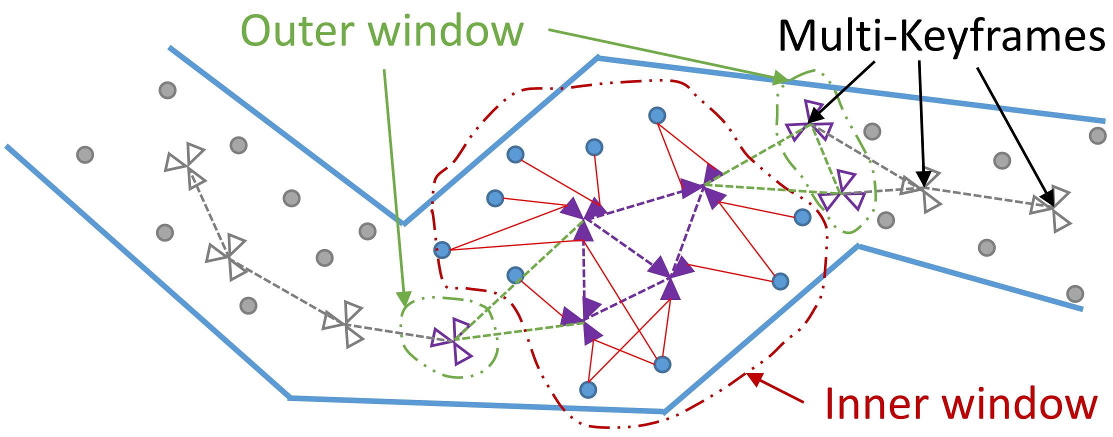


Figure 7.3. Principle of double window SLAM.

Seminal work [176] focused on keyframe optimization, selection and constraints. As the methods developed in [176] are basically used in ever modern keyframe-based SLAM, we describe them in more detail. Figure 7.3 depicts the double window approach. The inner window of active keyframes models the local area. Poses and map points are optimized using local BA. The outer window stabilizes the inner window and connects it to the rest of the trajectory. The question remains how to determine the connectivity between keyframes. In [176], the *co-visibility* is introduced. Instead of creating connections between keyframes based on geodesic or euclidean distance or temporal constraints, image features are used. By projecting map points to adjacent keyframes and matching their corresponding descriptors, a co-visibility weight can be calculated, that expresses how many equal features are visible in both keyframes. Apart from being able to update the connectivity if the scene changes, occlusions can be handled a lot better.

In [142], a complete SLAM system was proposed including loop detection, re-localization and handling of long-term dynamics. Co-visibility and local map querying was not only constrained by image to image matches, but also on the level of map points. A Histogram of Cameras (HoC) descriptor [141] is used to determine which map points are visible from the current camera pose. SIFT features are used and extraction and matching is performed on the GPU to achieve real-time performance, which limits its applicability on mobile platforms having only limited computing power.

In [110], the authors also build on the double-window optimization and co-visibility ideas but use the FAST detector coupled with BRIEF to exploit the performance of binary features. Using BRIEF and FAST decreases the time for feature extraction and matching significantly, however, this detector-descriptor combination is not rotation and scale invariant and thus limited to in-plane trajectories (like driving cars). In addition, points are only queried and tracked from the last keyframe. Thus, the local map structure is not fully exploited. ORB-SLAM [135, 136] is the latest state-of-the-art feature- and keyframe-based SLAM system, that is available on-line. As the name suggests, ORB features are used, being rotation and scale invariant to some degree. The map is reused and queried efficiently using co-visibility computed on ORB features. Place recognition is based on a binary BoW method [59] and a new map initialization heuristic is introduced, that dynamically switches between fundamental and homography estimation. Robust keyframe and map-point culling heuristics ensure a high map quality. We will build our multi-fisheye camera SLAM upon ORB-SLAM and explain all changes in the next sections.

The methods described so far either use a single camera (Monocular SLAM) or a stereo configuration (Stereo SLAM). CoSLAM (Collaborative SLAM) [219] aims at combining the maps build by multiple cameras moving around in dynamic environments independently. The authors introduce inter-camera tracking and mapping and methods to distinguish static background points and dynamically moving foreground objects. In [75], four cameras are rigidly coupled on a MAV. Two cameras are paired in a stereo configuration respectively and self-calibrated to an Inertial Measurement Unit (IMU) on-line. The mapping pipeline is similar to ORB-SLAM and also uses ORB descriptors for map point assignment. Additionally, the authors propose a novel 3-Point algorithm to estimate the relative motion of the MAV including IMU measurements. Most recent work on multi-camera SLAM is dubbed MC-PTAM (Multi-Camera PTAM) [69, 70] and is build upon PTAM. In a first step [69], the authors changed the perspective camera model to the generic polynomial model that is also used in this thesis (see Chapter 3). This induces further changes, e.g. relating the epipolar correspondence search that now has to be performed on great circles on the unit sphere instead of point to line distances in the plane. In addition, significant changes concerning the tracking and mapping pipeline had to be made to include multiple rigidly coupled cameras. Keyframes are extended to Multi-Keyframe (MKF)s as they now hold more than one camera. As PTAM, their system uses patches as image features and warps them prior to matching. Still, the system lacks a mapping pipeline that is capable to perform in large-scale environments. Subsequent work [70] improved upon [69] and is partly similar to the SLAM system developed in this thesis in that it uses the same camera model and g2o to perform graph optimization. On top, the authors integrated an automated calibration pipeline to estimate the relative orientation of each camera in the MCS (see also Chapter 4.4). Still the system uses the relatively simple mapping back-end of PTAM instead of double-window optimization that is used in this thesis and has proven to be superior. In addition, image patches are used as features making place recognition, loop closing and the exploration and storage of large environments critical.

Thus far, all approaches were based on local point image features. Hence, the reconstructed environment will stay relatively sparse even if hundreds of features are extracted in each keyframe. This makes it difficult for autonomous vehicles or robots that explore their surrounding to automatically analyze and extract object structure or texture information. Thus, most of the time, camera localization is coupled with laser scanners [111], structured light [85], yielding structured object information. Recent work on semi-dense [44, 52] and dense [34, 138] camera-based SLAM systems make use of a single camera to estimate dense scene structure instead of reconstructing only point features.

LSD-SLAM [44] is a semi-dense approach that runs on a single CPU in real-time, in contrast to dense methods [138] that need heavy GPU support. Using direct image-alignment by minimizing the photometric error between image discontinuities, the method skips the costly feature extraction and matching stage of all feature-based SLAM systems. The time saved compensates for the increased BA runtime, as a huge number of observations is included. In addition, all scale-drift aware loop closing and large scale double window optimizations are included, making LSD-SLAM a state-of-the-art approach that also runs in real-time. However, loop closing uses FAB-MAP [36] for place recognition and thus requires SURF features to be extracted. Subsequent work extended the method to mobile phones [167], stereo [45] as well as omnidirectional cameras [28]. Instead of coupling camera pose estimation and semi-dense mapping, in [137] a semi-dense extension to ORB-SLAM is presented. The semi-dense map is reconstructed from feature-based keyframes using depth consistency tests and a novel correspondence search. The semi-dense reconstruction is not obtained in real-time but is calculated in a CPU thread running in parallel to tracking and mapping. The methods yields superior performance compared to LSD-SLAM and it seems that the decoupling is advantageous, especially in dynamic scenes.

7.1.2 Model-based Tracking

In this thesis, geo-referenced, meshed 3D models of large construction sites are available. Exploiting the absolute spatial information for camera pose initialization, tracking and mapping would certainly help to improve long-term tracking and alleviate some drawbacks of the aforementioned relative methods, e.g. scale-drift.

Probably the first real-time model-based tracker was proposed by [71] and was called RAPID (Real-time Attitude and Position Determination). The method projects control points that are distributed along the edges of a known 3D model into the image. Then, the distance between those points and image edge points is minimized by optimizing the camera pose. As rather simple objects like boxes were used, occlusion handling was done by calculating view dependent look-up tables for edge points off-line. By probabilistic fusion of FAST features back-projected to the model surface and model lines, [151] present a high performance tracking method. The algorithm is capable of handling large camera shakes and rapid motion. A similar system was presented by [200, 201] but point tracking is performed on Harris corners. In [149] a system is presented that uses the modern graphics pipeline of OpenGL to extract salient model edges from a rendered image of a textured model. The advantage is that the composition of the level of detail of objects, silhouettes and texture is done by the renderer and a realistic image is composed. In doing so, the system was able to track large parts of a textured city model on a mobile platform with integrated IMU. Tracking industrial objects from untextured 3D models was proposed by [211]. Since texture is not available, model edges have to be created synthetically by rendering a normal and a depth view of the object, similar to Chapter 6. In [89], full edge tracking is performed on the GPU using a Particle Filter. Multiple edge images are rendered from sampled camera poses (particles) using OpenGL. The likelihood of each particle to be the correct pose is calculated by counting correctly matched edge pixels with the real camera image. The authors of [10] propose a method that is very similar but works for more complex models albeit still being small. The approaches presented so far initialize the tracking by hand. A robust method for the automatic detection of textureless 3D models in images was presented by [172, 194] and [77]. Again, normal and depth images are rendered, to extract all visible model edges. Then the cosine similarity (cf. Equation 6.2) is computed for multiple instances of the rendered object. In [77] the object is not only detected but also tracked afterwards. Recent similar approaches are based on learning deep convolutional neural networks for 3D object detection, recognition and pose estimation [98, 206] giving state-of-the-art performance on related datasets. However, these approaches are designed to detect workspace sized objects that fully fit into the camera image. Even most of the follow-up work on model-based edge tracking like [30, 132, 187] basically use particle filtering and rendering of the object with frame rate and are thus limited to smaller objects.

In recent years, less effort was made to design edge-based tracking of CAD like objects, as they have to be provided and created somehow in advance. Structure sensors made it possible to create small models from scratch and use them directly for camera pose tracking. Slam++ [156] performs real-time multiple object recognition and tracking from a hand-held depth camera. Using SLAM techniques, subsequent optimization of the pose to object graph allows to build a consistent object map and stable camera pose.

7.1.3 Open Challenges and Contributions

Tracking untextured objects of the size presented in this work, with a large extend and vertex count, will potentially be hardly possible at frame rate for a MCS, even in near future. Thus, we propose to combine the advantages of model-based tracking methods (cf. Section 7.1.2) and the robust map building that is part of keyframe approaches (cf. Section 7.1.1) of recent SLAM algorithms. First, we will extend the state-of-the-art ORB-SLAM to multi-fisheye camera systems using MultiCol (cf. Chapter 4). Instead of employing image patches as features (cf. MC-PTAM), compact binary descriptors proved to be the state-of-the-art when it comes to efficient large-scale re-localization, tracking and loop closing. As most binary features that are used in state-of-the-art SLAM systems are only designed for perspective cameras, we include online learned and distorted binary descriptors (mdBRIEF) as proposed in Section 5.

An additional novelty of our approach is to integrate the spatial information provided by the planned 3D models into the SLAM pipeline. By using modern graphics techniques, we render views of the model from all MKF poses. Then, line features that are pre-sampled from the model edges are projected into the MKF images. Using the rendered depth images of each MKF, hidden-line removal is performed and the model edges are aligned to edges extracted from the camera images. Instead of performing edge-based tracking at frame rate, the camera pose between MKFs is estimated from points that are triangulated across MKFs. These points are additionally analyzed and classified into being part of the model or not.

7.2 Framework

In this chapter, the proposed SLAM system is introduced. As mentioned previously, the basic structure of our system is build upon ORB-SLAM [136]. The proposed tracking and mapping system is dubbed MultiCol-SLAM. Figure 7.4 depicts an overview of the system. In general the system is divided into multiple threads running in parallel and taking care of different aspects. For the sake of clarity, the loop detection thread is omitted in this figure. Each adaption to the original ORB-SLAM system is highlighted in red and will be explained in more detail in the following. Two of the most profound adaptions in MultiCol-SLAM compared to ORB-SLAM is the introduction of Multi-Keyframes (MKFs), i.e. a keyframe consists of multiple images and the use of fisheye cameras. Both novelties involve some significant changes to the basic design, e.g. bundle adjustment, pose estimation, map point triangulation and relative orientation computation.

With every new set of incoming camera images, the tracking thread extracts point features from every image. Then, they are stored in a continuous vector, that will later be used to identify and match points across MKFs and mask outliers. To ensure a fast indexing and querying of feature to camera mappings, we use hash maps (called unordered maps in C++) that provide constant time ($O(1)$) search. Like ORB-SLAM, we use the relative orientation between the last two frames to predict the current position of the system. The local map points are projected to the MCS and matched to the extracted features from the current frame. If enough matches are retained from the set of putative correspondences after an initial robust pose optimization using MultiCol, the tracking thread starts to search for more matches, assigns the reference MKF and decides if a new MKF should be added and passed over to the mapping thread. If the initial pose estimation fails, GP3P [92] and RANSAC are used to perform re-localization of the MKS using the map points assigned to a set of recent MKFs. This is different compared to ORB-SLAM where a single camera and non-minimal PnP solver (EPnP [105]) is used in a RANSAC loop to find potential map point matches and estimate the current camera pose after tracking failure. The tracking thread is detailed in Section 7.2.2.

Each time the tracking thread passes a new MKF to the mapping thread, recently created map points that do not fulfill certain conditions are deleted from the map. Then, new map points are triangulated over MKFs that are in the vicinity of the added MKF. Here, the vicinity is determined by the co-visibility graph. In contrast to ORB-SLAM, where features are only triangulated between images of the same camera, the reconstruction is now performed over images of different cameras as well. Subsequently, local bundle adjustment is performed to adjust the poses of the MKFs that are part of the local map, as well as all map points. In addition, the mapping thread decides which MKFs are redundant and deletes them from the map. Another novel feature of MultiCol-SLAM is the introduction of online descriptor mask learning (cf. Section 5.4.2). ORB-SLAM assigns to each map point a distinct descriptor whose Hamming distance is minimal with respect to all other descriptors that are associated to it in different keyframes. Instead, we learn a mask that represents the binary tests that are robust to viewpoint changes. This is a lot faster than matching multiple descriptors of the same keypoint to find a distinct representation. The mapping thread is detailed in Section 7.2.3.

The loop closing thread searches for potential loop closures with every new MKF that is added. To decide if a place was already visited before and to identify MKFs as loop candidates, the system uses a Bag-of-Binary-Words framework [59]. In contrast to ORB-SLAM, we learn the visual vocabulary on dBRIEF features. If a loop is detected, the essential graph (a sparse version of the co-visibility graph) is optimized. To correct for the scale-drift, the optimization is carried out over similarity transformations that connect the MKFs.

Like ORB-SLAM, we use the graph optimization framework g2o [102] for all optimizations. The difference to ORB-SLAM is how we model the tracking and mapping pipeline. As multiple cameras observe the scene from each pose (Figure 7.2) and also one map point can be observed by multiple cameras at the same time, the graph can not be represented by binary edges anymore (edges that connect two vertexes). Instead, we extend the graph to a hyper graph that we used to model MultiCol (cf. Figure 4.2) where edges can connect to an arbitrary number of vertexes.

In the next section, the basic entities and methods that are used to represent the environment, i.e. map points, Multi-Keyframes, co-visibility graph and the 3D model are detailed.

7.2.1 Map Entities

Map Points

The map point is the most basic entity of the framework. Each map point $\mathbf{p}_i$ has the following attributes, properties and variables:

- The point class $C \in C_m, C_s, C_e$ that groups points into one of three categories. Points of the first class C_m are potentially part of the 3D model. The second class C_s of points does not lie on or close to the model surface but is extracted while tracking is performed within the 3D model. In addition, those points have more than 4 observations across MKFs and are thus considered to be reconstructed well and stable. Still some of those points might vanish in future inspections. The third class C_e of points are created if the system performs full exploration, i.e. the current system pose is likely to be outside the model boundaries.

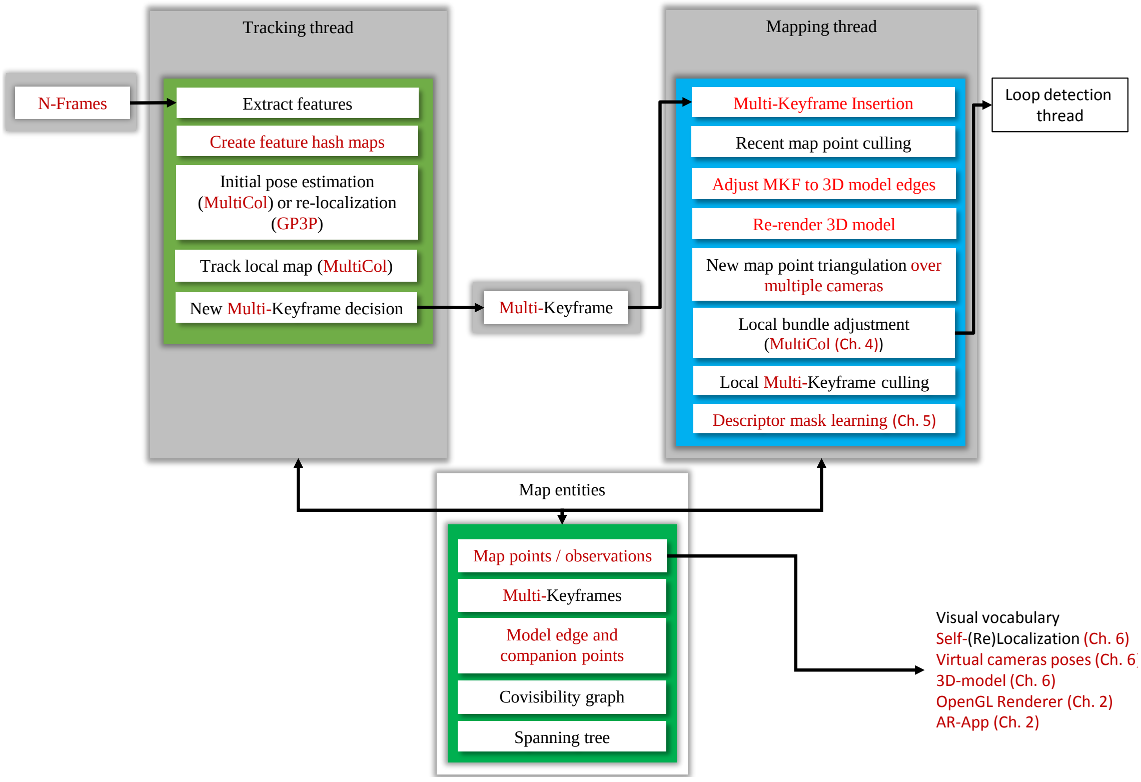

Tracking thread
N-Frames
Extract features
Create feature hash maps
Initial pose estimation (MultiCol) or re-localization (GP3P)
Track local map (MultiCol)
New Multi-Keyframe decision
Multi-Keyframe
Mapping thread
Multi-Keyframe Insertion
Recent map point culling
Adjust MKF to 3D model edges
Re-render 3D model
New map point triangulation over multiple cameras
Local bundle adjustment (MultiCol (Ch. 4))
Local Multi-Keyframe culling
Descriptor mask learning (Ch. 5)
Loop detection thread
Map entities
Map points / observations
Multi-Keyframes
Model edge and companion points
Covisibility graph
Spanning tree
Visual vocabulary
Self-(Re)Localization (Ch. 6)
Virtual cameras poses (Ch. 6
3D-model (Ch. 6)
OpenGL Renderer (Ch. 2)
AR-App (Ch. 2)

Figure 7.4. Depicts the MultiCol-SLAM framework without the loop closing thread. Red text depicts the modules where significant difference between our method and ORB-SLAM exist.

- The 3D position $\mathbf{p}_i = [X_i, Y_i, Z_i]^T$ in world coordinates.
- A maximum d_{max} and minimum d_{min} distance at which the point can be observed. This distance is used to reduce the number of points that are queried for local map tracking.
- The viewing direction $\mathbf{n}_i = [n_x, n_y, n_z]^T$. If the point is of class one we set $\mathbf{n}_i$ to be the normal vector derived from the 3D model. Otherwise the normal vector is calculated by averaging over the view directions from the MKFs to the map point.

Multi-Keyframes

In contrast to ORB-SLAM, each keyframe stores multiple images and is thus called Multi-Keyframe. Again, each Multi-Keyframe MKF_t created at time t has a number of attributes and variables:

- The rigid body MCS pose $\mathbf{M}_t$ (see also Equation 4.18).
- A MCS object that stores the intrinsics of each involved camera and the extrinsics of the MCS. This object also performs the forward and back projection of world and image points.
- All features that are extracted from each camera. The features are stored in continuous vectors and thus a fast feature to camera search is needed. For each image point, we store two representations. One is its 2D image coordinate $\mathbf{m}'$ that we use extensively in MultiCol bundle adjustment and pose estimation. In addition, we store the corresponding 3D bearing vector $\mathbf{v}$. Latter will be used in various geometry related algorithms, e.g. essential matrix estimation, epipolar search and absolute pose estimation (GP3P).
- The Bag-of-Binary-Word representation.

The following attributes are only used if model-supported tracking is performed:

- If the MKF is inside the model, we render three images (normal, depth and world coordinates) for each camera, from which map point coordinates $\mathbf{p}_i$ and viewing directions $\mathbf{n}_i$ can be extracted.

- Canny edge and orientation images.
- A list of model edge and companion points $\mathbf{p}_e$ and $\mathbf{p}_{co}$, their projections $\mathbf{m}'_e$, $\mathbf{m}'_{co}$ in the images and the model edge orientation ϕ_e at $\mathbf{m}'_e$. The purpose of these points will be detailed in Section 7.2.4. In addition, we will store a corresponding image edge point $\mathbf{m}'_{eI}$, that will be determined in the mapping thread.

Co-Visibility Graph

As in ORB-SLAM, the co-visibility graph is represented as a weighted undirected acyclic graph. The weight χ of each edge that connects two nodes (MKFs) in the graph is calculated as the number of map points the two MKFs share. In [176], a minimum weight χ_{min} between 15 and 30 was used to insert a connection. In contrast, the authors of ORB-SLAM do not impose a constraint on the minimal weight. When needed, the co-visibility graph is queried with a threshold, to only return nodes with a weight larger than χ_{min}, but the connectivity is kept very dense.

Map Initialization

Although map initialization plays a minor role in this context, as the initial pose is supposed to be initialized from the 3D model, we analyze possible methodologies for the initialization of the system without model information.

In general, the initialization of the map is the first crucial step in camera based SLAM system. The accuracy and robustness of the initial reconstruction has a significant impact on the overall performance of the system. Again, similarities to geodetic traverses are obvious, where mistakes in the initial orientation and position affect the whole trajectory significantly.

The authors of ORB-SLAM introduce an algorithm that switches between scene reconstruction using the fundamental matrix $\mathbf{F}$ or estimating a homography $\mathbf{H}$ using a heuristic. In general, this is a challenging task and often ignored, assuming the first camera motion introduces enough parallax. If a homography describes the current scene better, i.e. if the camera is only rotating or observes a completely planar scene, initialization is suppressed. If a proper fundamental matrix is found, the scene is initialized, by reconstructing camera poses and scene points, followed by bundle adjustment eliminating the gauge freedom by fixing the first camera.

With MultiCol-SLAM, two issues arise that require some adaption. On the one hand, both $\mathbf{F}$ and $\mathbf{H}$ matrices contain the perspective camera matrix $\mathbf{K}$. For omnidirectional or fisheye cameras and especially the camera model employed in this work, a camera matrix does not exists. Still, if images points would be undistorted to their corresponding location in a perspective image, an application would be possible, although points at the image border would be lost. On the other hand, we are actually looking for the relative motion between two MCSs. Thus a map has to be initialized for each camera separately and then fused somehow. A different approach is to directly estimate the relative orientation between two MCS poses which is equivalent to computing the relative pose between two generalized cameras [213]. We experimented with the linear 17-pt [213], a 6-pt [174] and a 8-pt algorithm [93] all part of OpenGV. The two polynomial solvers are relatively slow and the linear 17-pt algorithms is numerically very unstable. Recently, a new method was published [202] but we leave the investigation for future work.

As the initial reconstruction of the SLAM trajectory is not at the core of this work, we propose a rather practical than generic methodology. We estimate an essential matrix $\mathbf{E}$ in a RANSAC loop between the same camera from different MCS poses and choose the one with the most inliers and the highest translational magnitude. Then, we exploit the slight overlap between the FoVs of the helmet camera system and search for the map point projection in all other cameras. Finally, we perform bundle adjustment over all observations and the two camera poses.

7.2.2 Tracking Thread

In this section, the tracking thread is detailed. It is the core of our multi-camera tracking system as it handles not only the current state but performs feature extraction, matching and pose estimation. At the same time, it is the only thread that has to run in real-time, i.e. at frame rate. If this is not the case, incoming camera images will be dropped and tracking will suffer. Thus, an efficient implementation of all methods is essential. In addition, the tracking thread handles the MKF insertion and takes care of distributing work to all other threads.

All optimizations are carried out using iterative re-weighted least squares (IRLS) using Levenberg-Marquardt regularization and a robust Huber kernel (Section 2.1). Huber suggested to calculate the tuning constant e as $e = 1.345\sigma$, where σ is some estimate of the standard deviation of the residuals, that we set to $\sigma = 2$. After each optimization, outlier edges (measurements) are found and eliminated by testing the residual against the Huber value.

Feature extraction

The standard ORB detector extracts FAST corners at multiple scale levels (usually 8) and retains a certain number of corners per level that fulfill the Harris cornerness measure. As FAST corners usually need a re-training in new environments, we chose to utilize AGAST corners instead. Moreover, in ORB-SLAM each image is divided into several cells on each pyramid level. Then, the extractor tries to find at least 5 corners per cell and a total of N=1000 corners per image. If this is not the case the, cornerness threshold is adapted. Finally, the feature orientation and a ORB descriptor is computed.

In MultiCol-SLAM, we extract dBRIEF or mdBRIEF descriptors (cf. Section 5) on each AGAST feature to account for the fisheye image distortion. If mask learning is activated (mdBRIEF), a descriptor mask is additionally learned for each feature. If masks are available for each feature the *bilateral* masked Hamming distance defined in Equation 5.14 is calculated whenever descriptors are matched.

We will evaluate the impact of these changes in the experimental section.

Tracking from the Previous Pose

This step is similar to ORB-SLAM. First, the current pose is estimated by using a constant velocity motion model. Then, local map points assigned to the last pose are projected to each camera of the current MCS and a guided search is performed around the projected location. With this initial set of matches, the MCS pose is optimized using MultiCol on fixed map points and outlier measurements are marked by identifying edges with residuals over e. With the optimized pose, the guided search is repeated, to identify more potential matches and the pose is optimized again.

Re-Localization

As soon as the tracking thread indicates a tracking failure, re-localization is carried out. This happens if not enough points are retained after the initial pose tracking. Then, the images are converted into their corresponding Bag-of-Words representation, and the recognition database is queried for potential MKF candidates. We iterate through each MKF and match all associated map points to the keypoints detected in the current frame yielding a set of putative correspondences. In ORB-SLAM, the initial pose estimate is found using EPnP in a RANSAC loop.

We exchange EPnP for two reasons. On the one hand, EPnP requires more than three points to estimate the current pose of the camera and is thus not a minimal solution. The EPnP estimation is furthermore rather unstable for only few points. On the other hand, as we build our system on a MCS, we try to solve for the six degrees of freedom of the MCS pose using observations from multiple cameras. Thus, GP3P+RANSAC [92] is used to find a putative set of inliers for all MKF candidates. If enough inliers are retained and RANSAC did not exceed a predefined number of iterations, we refine the initial pose estimate obtained by GP3P for each MKF over all inliers using GPnP [92]. Finally, the pose is optimized with MultiCol, again suppressing map point correspondences with high residuals. If more than a predefined number of points (we set this to 15) is retained after final pose optimization, re-localization was successful and the tracking thread goes back into its usual behaviour.

Tracking the Local Map

The final step of the tracking loop is important, as it reinforces the visual connectivity and builds a densely connected co-visibility graph. Having estimated an initial pose either from tracking the previous pose or after re-localization, the local map is projected to the MCS to find more matches. To identify which map points are contained in the local map, a reference MKF to the current pose has to be found first. Thus, we take the list of map points that are currently assigned to the MCS pose entity and from which the current pose was estimated. As each map point stores a list of MKFs it has been observed in, we can then iterate through all map points and count their occurrences. Care has to be taken, as each map point can be observed multiple times from each MKF. The MKF with the most occurrences is taken as the reference MKF and a set of local MKFs is queried from the co-visibility graph.

Then, the following steps are performed consecutively over each point $\mathbf{p}_i$ in the local map:

(1) Project $\mathbf{p}_i$ into each camera of the MKF. Discard if projection is not inside the mirror boundary of a specific camera. Otherwise add the point to potential candidates.

(2) Compute the angle between the current bearing vector $\mathbf{v}_i$ and the map point viewing direction $\mathbf{n}_i$ and discard if angle is larger than $50°$.

(3) Compute the distance d_i from the current MCS pose to the map point. If this distance is outside the interval d_{min} and d_{max} that are defined by the scale invariance region of the image pyramid, discard the point.

(4) Get all descriptors around the projected location of the map point at a given scale and match them to the map point descriptor.

Finally, the pose is optimized for the last time. If not enough matches are retained, the tracking thread goes into re-localization mode.

New Multi-Keyframe Decision

From a robustly estimated MKF pose that is tightly connected to the local map and co-visible MKFs the tracking thread decides if it is time to add a new MKF to the map. The insertion takes place if the following conditions are met. All thresholds are set according to the Frames per Second (FPS) of the camera system (our MCS runs at FPS = 25):

(1) More than $0.5 \cdot$ FPS frames have passed from last MKF insertion and the local mapping thread is idle.

(2) A certain amount of poses must be successfully tracked from the last re-localization. In our case this threshold is set to the current frame rate.

(3) At least 50 points are tracked from the current MCS pose.

(4) Less than 90% of the current map points are assigned to the reference MKF. Thus, MKFs are only inserted if the visual change is big enough.

(5) A minimum distance between the current MCS pose and the reference must be exceeded. We roughly estimate this distance from the median scene depth δ_{med} and apply it after a couple of MKFs are already inserted into the map. Given the basis b between two cameras, their focal lengths c in meter and an estimate of the image measurement accuracy σ_m the reconstruction error in z direction σ_z can be approximated by $\sigma_z = \frac{\delta^2 \sigma_m}{bc}$. Thus the minimum distance between two MKFs can be roughly derived from a desired reconstruction quality. Usually we set $\sigma_z = 0.1m$.

In ORB-SLAM, the last condition is avoided. For MultiCol-SLAM where a 360° view of the environment is present at all time, we found it to be important, as to many MKFs where inserted and the reconstruction quality suffered.

7.2.3 Mapping Thread

This section details the mapping thread (cf. Figure 7.4). Asynchronously to the tracking thread, it extends the map by triangulating new points, performs local bundle adjustment and deletes redundant map points and MKFs from the map. Every time it finishes one loop, the entities are fed back to the tracking thread and become available. If a 3D model is available, the mapping thread additionally aligns each incoming MKF to the model.

Multi-Keyframe Insertion

As soon as the tracking thread decides to insert a new MKF into the map, the mapping thread starts to update the co-visibility graph. The last step in the tracking thread ensured a tight connectivity. Thus a new node is added to the graph and the edge weights are updated with the number of map points each MKF shares with the new MKF. In addition, a Bag-of-Words representation of the new MKF is computed and saved in the recognition database.

If the tracking takes places inside a 3D model, the following steps are performed:

(1) Render the depth images from the current MKF pose $\mathbf{M}_t$, that was estimated from tracking feature points.

(2) Extract image edges. For each image of the MCS, we first extract gradients using a 3×3 Sobel kernel. Subsequently, the derivatives in x and y direction are passed to a Canny edge detector [26] to get thin edges. For all remaining edge pixels, we then extract edge orientations using Equation 6.1.

(3) Project all model edge and companion points to the MCS and perform a simple depth test using the rendered depth image. We add a small offset of 5 cm to the depth values, in order to compensate for uncertainties in the MCS pose. For very large 3D models, the model edge points could be queried from a kdtree with a radius search, to decrease the number of projection and test calculations.

(4) Estimate the orientation ϕ_e of the projected model edges that passed the depth test in the image using their projections $\mathbf{m}'_e$ and $\mathbf{m}'_{co}$.

(5) Search perpendicular to ϕ_e for image edges in both directions for t pixels. An image edge is found if the orientation difference is below some threshold t_e. Again, we compare edge orientations using Equation 6.2, to avoid polarity issues. As soon as an edge is found in one direction, we stop the search in that direction.

Recent Map Point Deletion

The mapping threads stores a list of recently added map points from the last three MKFs. All map points under consideration have to pass the following conditions to remain in the map. The conditions do not only take visibility in subsequent MKFs into account but also the appearance between them.

(1) A map point has to be found in at least 25% of its predicted MCS poses.

(2) The map point must be observed from at least three MKFs.

As in ORB-SLAM, a map point can only be removed from the map if it is visible in less than three MKFs. This can happen if MKFs are removed from the map.

Adjust MKF to Model

If tracking is performed inside a 3D model, we adjust the MKF to the model. After MKF insertion, we have pairs of model edge points $\mathbf{p}_e$ and corresponding image measurements $\mathbf{m}'_{eI}$. Thus, we have narrowed down the problem of aligning the MKF to the model edges to pose optimization. After adjusting the MKF pose, we re-render the model. This time, not only the depth, but also normal and world coordinate images are saved.

New Map Point Triangulation over Multiple Cameras

Map points are created by triangulation from MKFs. First the co-visibility graph is queried for five MKFs from the current reference MKF. Then the following iteration is carried out to create new map points:

(1) Check if the baseline between the reference and the current neighboring MKF is big enough and the reconstruction quality σ_z can be guaranteed.

(2) Search image points for triangulation. First the Essential matrix $\mathbf{E}$ is calculated for each camera pair. In this case of three cameras, we get nine essential matrices in two MKFs. Those will be used to verify that matched points lie on the corresponding epipolar great circle.

(3) Take all descriptors from the reference MKF and match them to all descriptors from the queried MKF.

(4) Discard, if point is too far from epipolar great circle or Hamming distance is above threshold.

If model-supported tracking is performed, we additionally do the following:

(5) Get the world coordinate images of each MKF and extract two world points $\mathbf{p}_{MKF_i}$ and $\mathbf{p}_{MKF_j}$ for each match. If the distance between both world points is below a threshold, we directly assign the newly created map point to class C_m, set the map point coordinate $\mathbf{p}_i$ to the mean world point coordinate of $\mathbf{p}_{MKF_i}$ and $\mathbf{p}_{MKF_j}$ and extract a normal vector $\mathbf{n}_i$ from the rendered normal images.

Finally map points are triangulated and tested for positive depth, parallax and reprojection error in both cameras from which they were triangulated. If the newly created map point passes all tests, it is added to the map, a descriptor mask is learned and it is passed to the recently added map point list, that in turn is used to cull bad map points.

Map point fusion

As the triangulation step might have yielded redundant points that were already in the map, the next step is to fuse those point duplications. Therefore, the map points connected to the current MKF are first projected to all MKFs that are connected in the co-visibility graph. Within each MKF the map points are projected to each camera. Then, all features in a local area around the projected point are queried and matched. If a match is found, that also lies on the epipolar great circle, the map points are fused. If the matched image point is not connected to a map point yet, it is added as an observation. After projecting from the current to all connected MKFs, the same procedure is carried out vice versa.

Local Bundle Adjustment

The local bundle adjustment optimizes over poses and map points in the inner window (cf. Figure 7.3). Therefore, we query the co-visibility graph from the current reference MKF to get a set of MKFs. Then, all map points that are seen from this set of MKFs are added to the local map. In addition, the outer window is found, by looping over the current set of local map points and identifying MKFs that are not yet part of the local set of MKFs. This stabilizes the trajectory and connects it to the rest of the map.

Again the MultiCol equations are used, but this time we optimize over the local map points $\mathbf{p}_i$ and poses $\mathbf{M}_t$. Although hundreds of points and dozens of poses are subject to optimization, the bundle adjustment problem can be efficiently solved by exploiting the special sparsity structure of the Jacobian $\mathbf{J}$ and Hessian $\mathbf{J}^T\mathbf{J}$ respectively. The special structure comes from the fact that only point-pose but no point-point or pose-pose constrains exist and hence that the normal equations can be partitioned into:

$$\begin{bmatrix} \mathbf{J}_\mathbf{P}^T\mathbf{J}_\mathbf{P} & \mathbf{J}_\mathbf{P}^T\mathbf{J}_\mathbf{M} \\ \mathbf{J}_\mathbf{M}^T\mathbf{J}_\mathbf{P} & \mathbf{J}_\mathbf{M}^T\mathbf{J}_\mathbf{M} \end{bmatrix} \begin{bmatrix} d\mathbf{p} \\ d\mathbf{M} \end{bmatrix} = \begin{bmatrix} -\mathbf{J}_\mathbf{P}\mathbf{r}_\mathbf{P} \\ -\mathbf{J}_\mathbf{M}\mathbf{r}_\mathbf{M} \end{bmatrix} \tag{7.1}$$

where $\mathbf{J}_\mathbf{P}$ and $\mathbf{J}_\mathbf{M}$ are the Jacobian matrices w.r.t points and poses, $d\mathbf{P}$ and $d\mathbf{M}$ are the parameter updates and $\mathbf{r}_P$, $\mathbf{r}_M$ are the residual vectors. Equation 7.1 can the be efficiently solved using the Schur complement trick. For more details and insights on the subject, the reader is referred to [46]. Subsequently, the local map is updated and passed back to the tracking thread.

Care has to be taken if model-supported tracking is enabled. Then, the MKF poses have already been adjusted by aligning image and model edges. In this case, we only optimize over the local map points $\mathbf{p}_i$.

Estimate Map Point Class

After adjusting the local map points, we estimate their class affiliation. Recall that we would like to distinguish between three different classes, i.e. model point C_m, stable points inside the model C_s, and map points outside the model volume, reconstructed during full SLAM.

A map point is considered to belong to class C_s if it has more than 3 observations across MKFs and tracking takes place inside a model. If the distance between its reconstructed and adjusted world coordinate $\mathbf{p}_i$ and the coordinate extracted from the world coordinate image of the MKF, we set the class to C_m. The threshold is set to 5cm, i.e. the point has to be closer than 5cm to the model to be classified as model point. For larger scenes, a heuristic based on the pose and point uncertainties could be found.

Local Multi-Keyframe Culling

Like in ORB-SLAM, we delete a MKF from the map if any pair of MKFs shares more than 90% of the same map points. Instead of counting map point observations for each camera of the MCS, we count only the occurrence per MKF.

7.2.4 Integration of 3D Model Information

Thus far, the integration of the 3D model into our MultiCol-SLAM was only briefly mentioned in the mapping thread. This section details the extraction of model edge and companion points and the search for corresponding image edges.

The model is supposed to support the tracking and mapping chain of our SLAM pipeline and help to classify points that are likely to be stable over time. Such points could be used with a higher probability for re-localization. In addition, the model provides scale information and images, e.g. taken for documentation, can be readily geo-referenced. In the following, we assume that the MCS localized itself in the building model using the methods presented in Chapter 6, thus the pose $\mathbf{M}_1$ of the first MKF w.r.t the model is known.

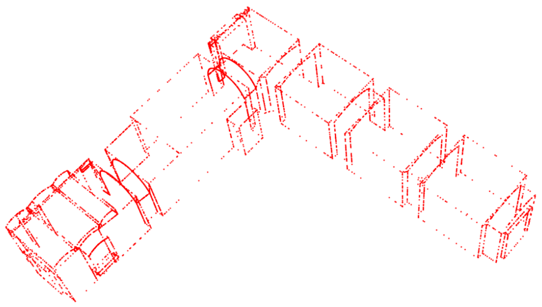
Different ways exist to integrate model information into the tracking process. One can either perform model-based edge tracking either in an independent pipeline or as a posterior refinement step of point-based frame to frame tracking. As an alignment of each MCS pose to the model edges was not real-time capable, we decided to align only MKFs to the model and thus support the trajectory estimation. Still, we use the pose of frame to frame point-based tracking as a prior.

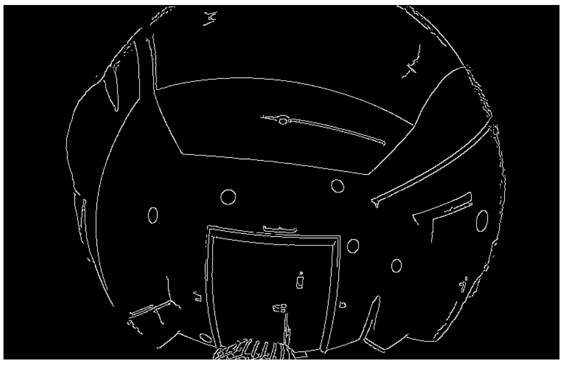
In general, the goal is to align 3D model edges to edges extracted in the MCS images of a MKF. First, we extract the model edge points $\mathbf{p}_e$ off-line, by analyzing the change of the normal vector between adjacent faces that a vertex is connected to. In this manner, we can identify vertexes that lie on a model edge. The extracted edge vertexes are depicted in Figure 7.5a. In addition, we extract a companion point $\mathbf{p}_{co}$ for each edge vertex. A companion point is the nearest vertex to $\mathbf{p}_e$ that also lies on the edge. This second point is needed to estimate the orientation of the model edge in the images.

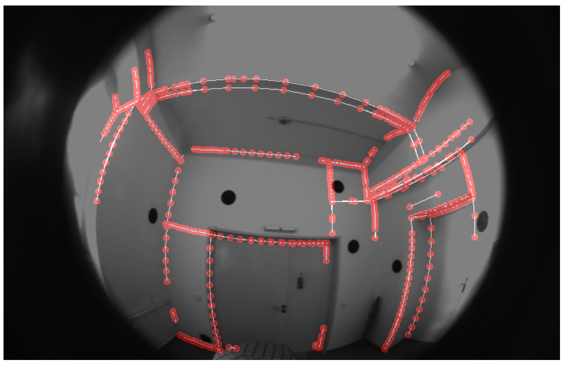
The image edges are extracted by first convolving each image of the MCS with a 3×3 Sobel kernel. Then the edges are thinned using Canny's method [26]. An example is depicted in Figure 7.5b; For each edge pixel that remains after non-maximum suppression, we finally calculate the edge orientation ϕ_e.

Then, for each new MKF, we project the model edge vertexes to the camera images. To retain only visible points, we use depth images that we render in advance. To account for the pose uncertainty, we add a small offset (5cm) to the depth values. Recall, that the initial pose estimate for the inserted MKF originates from point-based frame to frame tracking. Thus generally, the projected model points $\mathbf{p}_e$ will not exactly lie on the images edges. This is depicted in Figure 7.5c. Hence, we search along the edges normal direction for the corresponding image edge $\mathbf{m}'_e$. This is depicted in Figure 7.5d. The final measurements $\mathbf{m}'_e$ are depicted in green and the searched pixels are depicted by lines perpendicular to the projected model edges. An edge is only accepted if the orientation difference is below some threshold. The edge orientations are compared using the cosine similarity Equation 6.2.

Finally, the pose is optimized by minimizing the reprojection error of the edge points $\mathbf{p}_e$. The optimization is again carried out using MultiCol (Equation 4.18).


(a) Model edge points $\mathbf{p}_e$


(b) Canny edges


(c) Projected model points $\mathbf{p}_e$

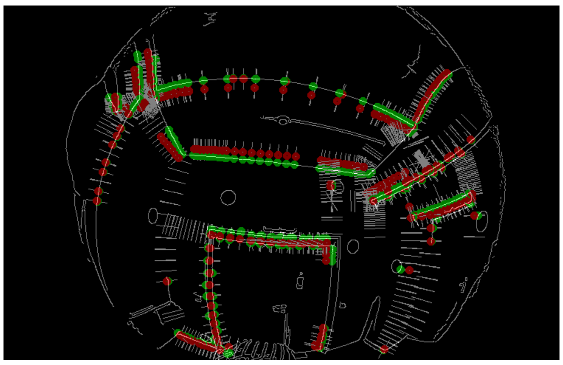

(d) Search lines and detected edge points $\mathbf{m}'_e$

Figure 7.5. Depicted is the process of detecting edge measurements for the model-supported tracking. (a) depicts the model edge vertexes $\mathbf{p}_e$. (b) depicts the Canny edge image of a real camera image. (c) depicts the projections of the map points $\mathbf{p}_e$ to the camera image using the MCS pose obtained by frame to frame point-based tracking. (d) depicts the search lines that are perpendicular to the edge directions. The edge directions ϕ_e are extracted using companion points $\mathbf{p}_{co}$ that are also projected to the image. The final edge measurements $\mathbf{m}'_e$ (green circles) are accepted if the angle between image edge and projected edge orientation is below some threshold.

7.2.5 Loop Detection and Closing Thread

In this section, the loop detection and correction procedures are detailed. They are similar to the methodology proposed in [136] but extended and adapted to multi-fisheye camera systems. As measurement errors accumulate over time, the estimated trajectory starts to drift in seven degrees of freedom, i.e. translation, rotation and scale. This effect is visualized in a toy example in Figure 7.6. Although the MCS visits the same place, the current local map (depicted in blue) does not coincide with the historic map (depicted in gray and orange). Although the map does not spatially align with the start of the trajectory, its local map structure is very similar assuming that the system reconstructs a large amount of map points at the same location, i.e. the reconstruction is repeatable. The goal of the loop detection thread is to identify the situation depicted in Figure 7.6, detect the loop candidates from the historic trajectory and correct the loop by propagating the loop closing error along the trajectory.

The loop detection and correction procedure, as well as the place recognition database and visual vocabulary components are depicted in Figure 7.7. Each step is detailed in the following sections.

Candidate detection

As soon as the system revisits a place or parts of a scene it has seen and reconstructed before, it should load the associated local map points and start tracking from them instead of starting to reconstruct the scene again. This, however, is a challenging task, as the local map is solely queried from the co-visibility graph. One possible way would be to query the map points spatially, i.e. that we query the nearest neighbor map points in a specified radius from the current MKF. This method could work for smaller loops, but as soon as larger loops occur, the corresponding local maps could lie dozens of meters apart, which is sketched in Figure 7.6. A more favorable solution is to use visual cues to identify possible loop candidates.

After local bundle adjustment, the current multi-keyframe MKF_i is handed over to the loop detection and correction thread. Now the MKF database could be directly queried for visually similar MKFs. However, the number of database results has to be limited somehow. This could either be achieved by taking a fixed number of MKFs sorted by their similarity score. Latter, however, is an absolute measure whose magnitude is unknown, e.g. we queried 10 MKFs, but all have a very bad similarity score. Thus a way of getting a relative measure of the quality of the current similarity score is needed.

Hence, the BoW vectors of all MKFs that are connected in the co-visibility graph of the current MKF_i are queried. Then, the similarity score between all BoW vectors and the current MKF BoW vector are calculated. The lowest score s_{sim} is taken as a similarity threshold, i.e. we only take MKFs from the database as candidates if their score is higher than s_{sim} and thus more similar, than the most dissimilar MKF in the co-visibility graph.

To further reduce the number of possible false candidates, MKF candidates are only accepted if they are part of a consistent group of connected MKFs in the co-visibility graph after several consecutive MKF insertions. Each loop candidate is connected to a number of MKFs (a group) in the co-visibility graph. A group is accepted to be consistent with the previous group if it they share a MKF.

Transformation estimation

If one or more candidates are accepted, the similarity transformation between the current MKF_i and all candidates can be estimated. Lets assume for now, that we have one candidate MKF_c. Looking back at Figure 7.6, the goal is to find the similarity transformation $\mathbf{S}$ (cf. Section 2.2.1.4) between the map points $\mathbf{p}^i$ assigned to MKF_i and the map points $\mathbf{p}^c$ assigned to MKF_c:

$$\mathbf{p}^i_{mcs} = \mathbf{S}\mathbf{p}^c_{mkf} = \begin{bmatrix} s\mathbf{R} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{p}^c_{mkf} \tag{7.2}$$

Instead of taking all map points that are connected to both frames, the descriptors assigned to each map points by the MKF are matched in advance. This leaves us with a subset of possible map point correspondences. Yet, this set contains outliers that are either caused by wrong descriptor matches or the distance ratios between reconstructed points is too big, caused by bad reconstruction. Hence,

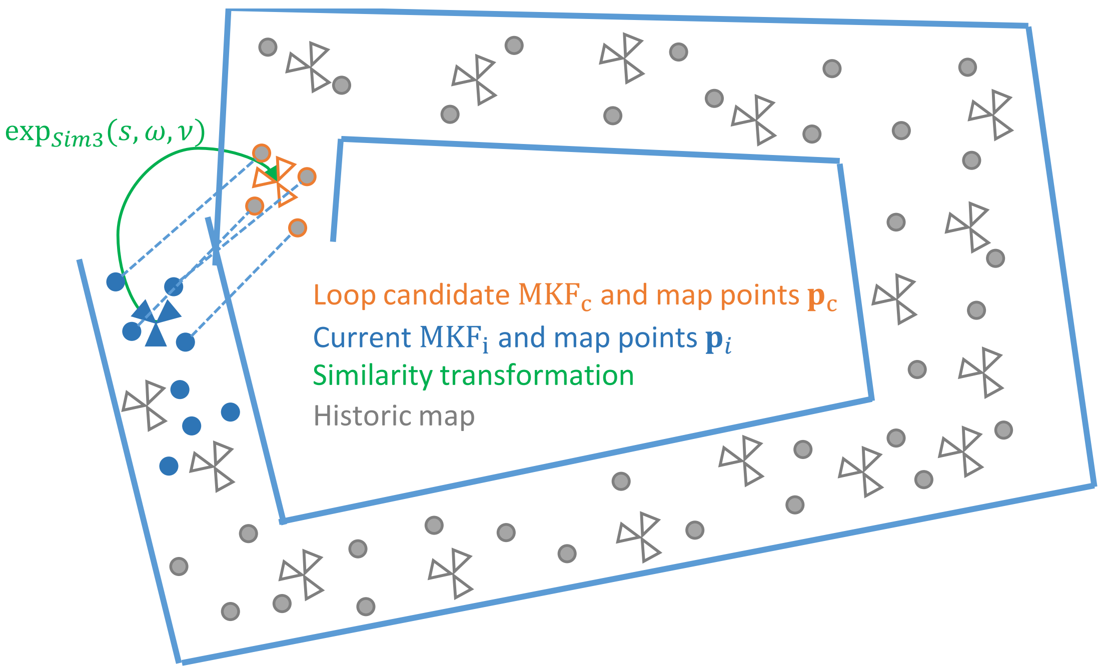

$\exp_{Sim3}(s, \omega, \nu)$
Loop candidate MKF$_c$ and map points $\mathbf{p}_c$
Current MKF$_i$ and map points $\mathbf{p}_i$
Similarity transformation
Historic map

Figure 7.6. Depicts the loop closing problem. If the SLAM trajectory was estimated without drift, the orange and blue map points should coincide. As this is in general not the case, a similarity transformation can be estimated that aligns both parts of the trajectory over the map points. Then, the alignment error can be used to correct the remaining MKF poses and map points by projecting it back through the map.

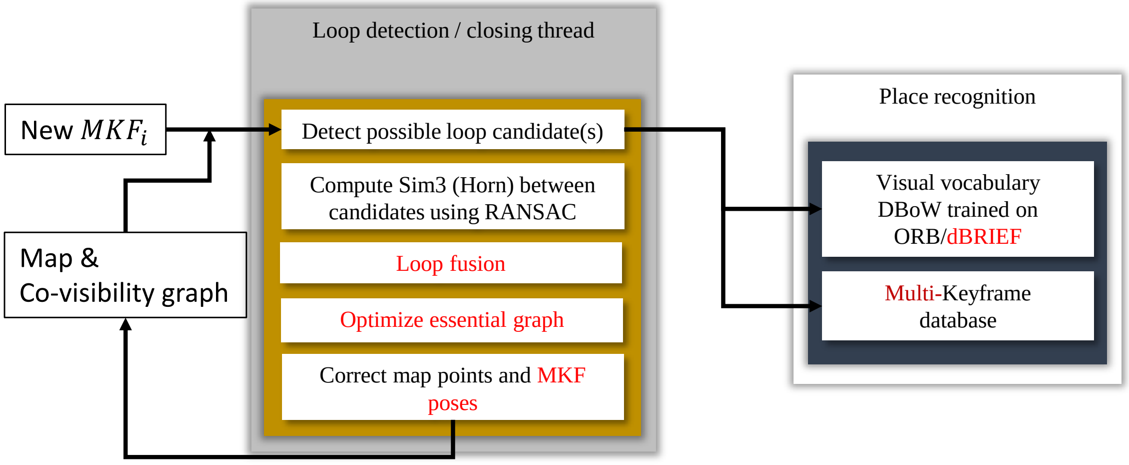

Loop detection / closing thread
New MKF_i
Map &
Co-visibility graph
Detect possible loop candidate(s)
Compute Sim3 (Horn) between candidates using RANSAC
Loop fusion
Optimize essential graph
Correct map points and MKF poses
Place recognition
Visual vocabulary DBoW trained on ORB/dBRIEF
Multi-Keyframe database

Figure 7.7. Depicts the loop detection and correction thread. It extends Figure 7.4. Each time a new MKF is inserted, the loop detection thread tries to detect possible loop candidates from the MKF database. First, a BoW score s_{min} is computed for all frames that are connected to the new MKF in the co-visibility graph. Then the MKF database is queried and only MKFs with a score higher than s_min are declared as candidates for loop detection. The right part shows the two components of the place recognition class.

RANSAC is used to find a similarity transformation using Horn's quaternion ([78]) method as a model and thus 3D-3D [1] correspondences.

First the set of map points matches is transformed to the respective MKF:

$$\mathbf{p}^c_{mkf} = \mathbf{M}^c_t \mathbf{p}^c \quad \mathbf{p}^i_{mkf} = \mathbf{M}^i_t \mathbf{p}^i \tag{7.3}$$

where $\mathbf{M}_t$ is the respective pose of the MKF. Obviously, the points can not be transformed to each camera of the MKF, as map points can be observed from multiple cameras and the only common frame is the MKF frame.

Subsequently RANSAC iterations are performed. Three points are selected from each point cloud, and the transformation is estimated. To decide whether the transformation is accepted, the map points are transformed from MKF$_c$ to MKF$_i$ and vice versa using the estimated similarity:

$$\hat{\mathbf{p}}^i_{mkf} = \mathbf{S}\mathbf{p}^c_{mkf} \quad \hat{\mathbf{p}}^c_{mkf} = \mathbf{S}^{-1}\mathbf{p}^i_{mkf} \tag{7.4}$$

Subsequently the points are transformed to the camera frames and projected to the image plane. Now, the reprojection error can be computed and used to determine the number of inliers. If the transformation yields enough inliers, a guided matching is instantiated to search for more correspondences, also between cameras. Then, $\mathbf{S}$ is optimized by minimizing the reprojection errors of both transformed point sets (Equation 7.4) in both MKFs. The optimization is again carried out using g2o over the minimal parametrization Equation 2.24 and outliers are down-weighted using a Huber kernel. If more than 20 inliers are retained after optimization, $\mathbf{S}$ is accepted and the loop correction is started.

Loop correction and fusion

The first step to loop correction is, to correct all MKFs that are connected to the current MKF$_i$, as well as all map points that are part of the local map. After this step, the local maps spanned by MKF$_i$ and MKF$_c$ should align. The corrected pose $\hat{\mathbf{M}}_t$ of a MKF is computed by first estimating the relative orientation between pose $\mathbf{M}^i_t$ of the current MKF$_i$ and the MKF pose $\mathbf{M}_t$ and subsequent correction using the similarity:

$$\hat{\mathbf{M}}_t = (\mathbf{M}_t \mathbf{M}^{i^{-1}}_t)\mathbf{S} \tag{7.5}$$

Subsequently, the map points need to be corrected as well. First, each map point is rotated to the MKF frame using the uncorrected pose. Then, the point is transformed to the corrected map point position $\hat{\mathbf{p}}$ by applying the inverse of the corrected MKF pose, i.e. the map point is directly transformed back into world coordinates:

$$\hat{\mathbf{p}} = \hat{\mathbf{M}}^{-1}_t(\mathbf{M}_t\mathbf{p}) \tag{7.6}$$

The correction of the local map will result in many point duplications and redundant MKFs. Thus the same map point fusion procedure presented in Section 7.2.3 is carried out and the co-visibility graph is updated.

Finally, the essential graph is optimized. First, all MKF poses $\mathbf{M}_t$ are converted to similarities $\mathbf{S}_t$ by initially setting the scale to 1.0. Then, the relative pose constrains between all MKFs in the map are computed:

$$\Delta\mathbf{M}_{ij} = \mathbf{M}_j\mathbf{M}^{-1}_i \tag{7.7}$$

between some MKF i and j. Again, all relative pose constrains are converted to similarities $\Delta\mathbf{S}_{ij}$. The optimization is carried out over pose-pose constrains and follows the work of [177]. The residual that we try to minimize is defined as:

$$\mathbf{r}_{i,j} = \log_{Sim(3)}(\Delta\mathbf{S}_{ij}\mathbf{S}_i\mathbf{S}^{-1}_j) \tag{7.8}$$

where the log is the inverse relation of the exponential map (see [175]). The goal of the optimization is to adjust $\mathbf{S}_i$ and $\mathbf{S}_j$ such that the transformation sequence (back and forth between both MKFs) is as close to the identity as possible. In the beginning, all residual transformations (cf. Equation

[1]Recently, new methods were proposed, that solve the generalized relative orientation and scale problem (similarity between MCSs) using 2D-3D [183] or 2D-2D [184] correspondences only to compute $\mathbf{S}$. Latter would alleviate the effects of reconstruction error, however, depends on an additional constraint, i.e. the current vertical direction needs to be determined. We leave the integration and investigation to future work.

7.8) will be the identity except for the part of the map, that was corrected above. Then the error is propagated back over all pose-pose constrains during optimization as the similarities $\mathbf{S}_i$ and $\mathbf{S}_j$ are gradually changed to optimally fit the loop closure constrain.

We optimize only the transformations between MKFs that are connected by at least 100 points, i.e. that the edge weight in the co-visibility graph is above 100. After optimization, all map points are corrected:

$$\hat{\mathbf{p}} = \hat{\mathbf{S}}_t^{-1}(\mathbf{M}_t \mathbf{p}) \tag{7.9}$$

using the corrected MKF pose $\hat{\mathbf{S}}_t$. and finally all $\hat{\mathbf{S}}_t$ are converted back to rigid body transformations:

$$\mathbf{M}_t = \begin{bmatrix} \mathbf{R} & \mathbf{t}/s \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{7.10}$$

where s is the scale.

7.3 Experiments and Results

To test the performance of the presented MCS SLAM, various tests are performed. First, we evaluate the impact of using multiple fisheye cameras instead of one, in terms of accuracy, runtime, successfully tracked frames and loop closing.

To evaluate the accuracy of SLAM systems, two metrics are commonly used that compare the estimated camera poses $\mathbf{M}_t$ to a ground truth pose $\mathbf{M}_t^{gt}$ at some time t or an interval Δt. The difference between these two poses at time t is given by the relative orientation between them:

$$\mathbf{M}_t^{rel} = \mathbf{M}_t^{gt^{-1}} \mathbf{M}_t \tag{7.11}$$

The first metric is called Absolute Trajectory Error (ATE) and estimates the root mean squared translation differences between both trajectories. In order to calculate the absolute error, the two trajectories need to be aligned in advance using a similarity transformation $\mathbf{S}$. For N pose pairs, the ATE can then be calculated as:

$$\text{ATE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \|\text{trans}(\mathbf{M}_t^{rel})\|^2} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} \|\text{trans}(\mathbf{M}_t^{gt^{-1}} \mathbf{S} \mathbf{M}_t)\|^2} \tag{7.12}$$

where "trans" returns the translational component of the transformation matrix $\mathbf{M}$.

The second metric is called Relative Pose Error (RPE) and allows to evaluate the local accuracy and drift of the trajectory over some time interval Δ. Thus we can calculate $M = N - \Delta$ relative orientation errors along the trajectory. The RPE at time step t can be defined by:

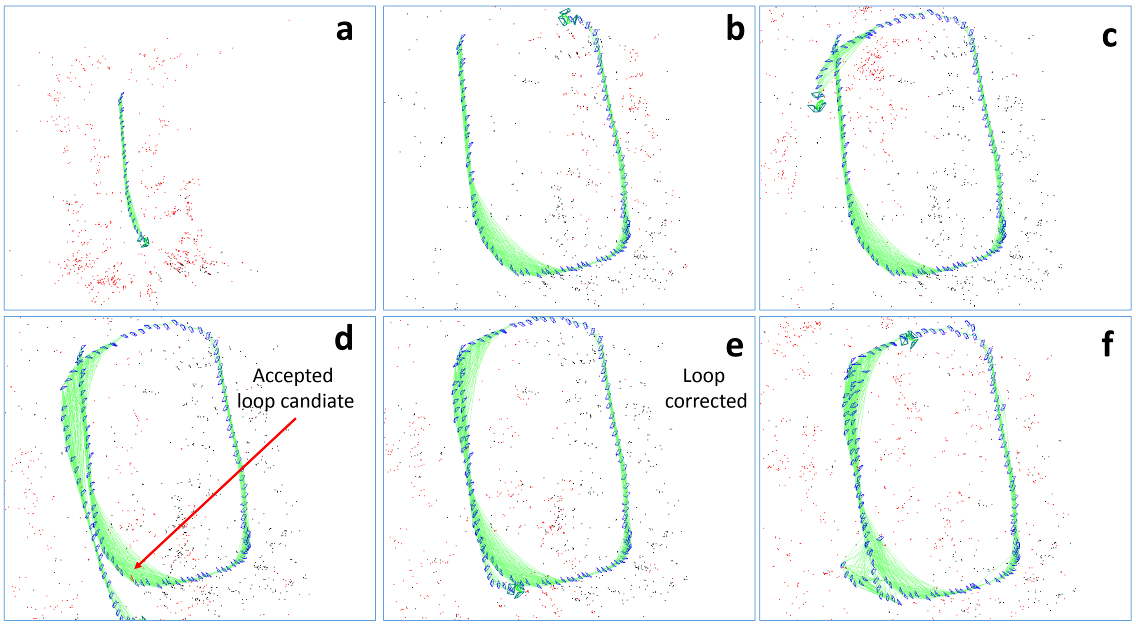
$$\text{RPE}(\Delta) = \sqrt{\frac{1}{M}\sum_{t=1}^{M} \|\text{trans}(\mathbf{M}_t^{rel})\|^2} \tag{7.13}$$

but this time the relative transformation is defined as:

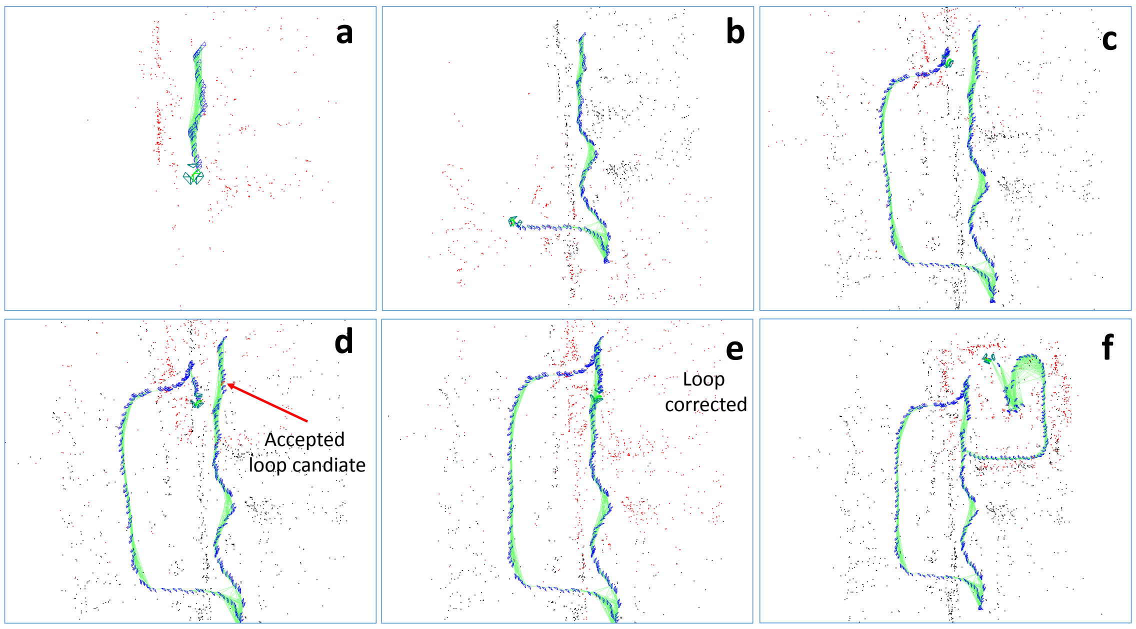
$$\mathbf{M}_t^{rel} = (\mathbf{M}_t^{gt^{-1}} \mathbf{M}_{t+\Delta}^{gt})^{-1} (\mathbf{M}_t^{-1} \mathbf{M}_{t+\Delta}) \tag{7.14}$$

To calculate the relative error of subsequent poses we set $\Delta = 1$. In the case of ATE only the translation is evaluated. For the relative error, we can also evaluate the rotational accuracy. This is done by replacing the "trans" with a function that returns the Rodriguez vector of the rotation matrix in $\mathbf{M}$.

Each trajectory is evaluated five times, i.e. the SLAM algorithms are used five times to estimate the camera trajectory. All accuracies and run-times are calculated as the median value over the five runs.



(a) Trajectory 1



(b) Trajectory 2

Figure 7.8. Depicted are two successful loop closures. **a-b** trajectory and map is build. The green lines represent the co-visibility graph edges with a weight higher than 100. Blue pyramids depict MKF. Red map points is the active map, black points are all map points. **c** Visually the trajectory crossed itself. However no loop is detected. Although candidates exist, RANSAC does not find a good solution, because the reconstructed geometry is still too different. **d** a loop candidate is accepted. **e** The corrected loop after the optimization of the essential graph. **f** tracking continues using the active map. New MKFs are only created if the baseline is large enough or to few points are tracked.

	Single fisheye camera			Multi-fisheye camera system		
	ORB	dBRIEF	mdBRIEF	ORB	dBRIEF	mdBRIEF
	[cm]	[cm]	[cm]	[cm]	[cm]	[cm]
Laser 1	31.0	33.0	33.0	1.4	1.3	1.5
Laser 2 fast	28.1	25.3	25.5	(X)	7.0	5.3
Indoor 1 stat. env.	32.4	11.2	3.3	2.1	2.7	2.5
Indoor 2 dyn. env.	13.3	14.7	14.2	1.8	2.5	2.7
Outdoor 1 dyn. env.	(X)	(X)	(X)	3.6	17.1	13.4

Table 7.1. Median KF and MKF **ATE**s for single and multi-camera SLAM respectively based on three different features. The translational accuracy was calculated after 7DoF alignment between ground truth and estimated frames. For trajectory details see Chapter 2.5.

	Single fisheye camera			Multi-fisheye camera system		
	ORB	dBRIEF	mdBRIEF	ORB	dBRIEF	mdBRIEF
	[cm]/[deg]	[cm]/[deg]	[cm]/[deg]	[cm]/[deg]	[cm]/[deg]	[cm]/[deg]
Laser 1	1.95/0.32	2.0/0.33	2.0/0.32	1.2/0.33	1.2/0.33	1.2/0.32
Laser 2 fast	2.6/0.31	2.4/0.31	2.5/0.30	(X)	2.7/0.52	2.7/0.51
Indoor 1 stat. env.	2.8/1.72	2.8/1.73	2.9/1.68	1.1/1.54	1.1/1.73	1.1/1.68
Indoor 2 dyn. env.	2.8/2.04	2.8/1.98	2.8/2.02	1.1/1.78	1.5/1.80	1.6/1.80
Outdoor 1 dyn. env.	(X)	(X)	(X)	2.3/1.28	2.8/1.48	2.9/1.28

Table 7.2. **RPE** for single and multi-fisheye camera SLAM. Here we set $\Delta = 1$, i.e. the frame two frame tracking accuracy is estimated. (X) means that tracking failed at some point and a significant part of the trajectory was not tracked. For trajectory details see Chapter 2.5.

7.3.1 Single- vs. Multi-Camera SLAM

First, we align the KFs or MKFs respectively, by estimating a similarity transformation between ground truth and SLAM trajectory. Then, the ATE (Equation 7.12) is evaluated for all trajectories. The results are depicted in Table 7.1. Obviously, MultiCol-SLAM significantly outperforms its single camera pendant in terms of Keyframe accuracy. One explanation of the large performance gap is the simple initialization of the single camera SLAM. The authors of ORB-SLAM proposed an initialization based on homography and fundamental matrix estimation. Both matrices can not be readily computed for the camera model employed in this work. Thus we simply initialize the single camera SLAM by estimating the essential matrix and selecting a solution based on a threshold on the magnitude of the translation vector, which is obviously less robust than the method proposed in [136].

To get a measure of the local accuracy, we also estimate the RPE (Equation 7.13) for all trajectories and all poses by setting $\Delta = 1$. The trajectories do not need to be aligned in this case. The accuracies for translation and rotation are depicted in Table 7.2. Still, using multi-cameras yields a better performance, especially for the translation. The rotational components show a similar trend, but the differences are less prominent. The rotational accuracy for the two lasertracker trajectories is a lot better because the trajectory has only little rotation of the camera system about the up-axis. On average the rotational accuracy of the MCS is about 1.0-1.5° and the translational component, depending on the walking speed and scene between 1.0-2.5 cm.

	Single fisheye camera			Multi-fisheye camera system		
	ORB	dBRIEF	mdBRIEF	ORB	dBRIEF	mdBRIEF
	% tracked	% tracked	% tracked	% tracked	% tracked	% tracked
Laser 1	88.4	89.7	89.7	88.6	88.7	88.7
Laser 2 fast	86.7	92.9	93.7	69.8	93.2	93.6
Indoor 1 stat. env.	95.2	91.5	96.9	95.5	98.8	98.7
Indoor 2 dyn. env.	91.8	95.1	95.1	97.5	98.7	99.0
Outdoor 1 dyn. env.	31.0	32.0	34.1	93.0	93.0	90.1

Table 7.3. Mean percent of successfully tracked frames, i.e. number of frames divided by the total number of available frames in the trajectory. For trajectory details see Chapter 2.5.

		Single Camera			Multi Camera		
Thread	Operation	ORB	dBRIEF	mdBRIEF	ORB	dBRIEF	mfBRIEF
Tracking	Frame creation	12/12/2	20/21/5	36/37/10	21/23/5	29/25/6	46/45/7
	Pose estimation	4/4/1	4/4/1	4/4/1	3/4/3	1/3/5	1/4/4
	Track local map	5/5/2	5/4/1	5/4/1	4/5/4	3/6/6	3/7/5
	Total	**21**	**29**	**45**	**28**	**33**	**50**
Mapping	Map point creation	27/26/10	32/36/18	51/56/20	96/96/17	85/83/17	94/94/18
	Map point fusion	11/11/6	9/11/6	13/15/9	11/12/9	14/14/8	17/20/10
	Local BA	185/198/125	156/200/141	182/256/182	170/174/65	224/214/61	296/279/81
	Total	**223**	**246**	**223**	**277**	**323**	**407**
Rendering	Depth images	-			8/8/1		
	Edge extraction	-			35/35/4		
	Edge features	-			34/34/7		
	Adjust model 2 MKF	-			127/181/38		
	Render all images	-			105/100/12		
	Total	~100			**309**		

Table 7.4. Depicted is the time for each step in different threads. For each descriptor the mean, median and standard deviation of the execution time is given. (mean[ms]/median[ms]/std[ms]). In case of Single camera SLAM, we extracted 1000 features. In case of multi-camera SLAM 400 features per camera are extracted and the extraction is performed in parallel. The pose estimation is slower in the single camera case, as no analytical Jacobian was provided. We did not evaluate the timings of the single camera rendering. This calculations, however, are not parallelized across camera, i.e. the total amount that is needed for the MCS can be divided by the number of cameras to get a rough estimate of the single camera timings (here about 100ms).

	ATE for MKFs			**RPE all frames**		
	Multi-fisheye camera system			Multi-fisheye camera system		
	ORB	dBRIEF	mdBRIEF	ORB	dBRIEF	mdBRIEF
	[cm]	[cm]	[cm]	[cm]/[deg]	[cm]/[deg]	[cm]/[deg]
Laser 1	3.83	3.7	2.60	1.6/0.61	1.5/0.53	1.2/0.53
Laser 2 fast	3.95	2.14	2.45	2.8/0.54	2.9/0.59	1.9/0.54

Table 7.5. Median MKF **ATE**s and median RPEs for all frames for multi-camera **model-supported** tracking based on three different features descriptors.

7.3.2 Impact of Features

To evaluate the impact of the binary features developed in Chapter 5, we perform several experiments. As depicted in Table 7.1 and Table 7.2 the features do not have a significant impact on the accuracy of the system. First of all, the features use the same detector principle (FAST or AGAST respectively). Thus the distribution of point in the images and the point localization accuracy is similar. In addition, the standard ORB descriptor already leaves us with many points to track and the geometric verification and guided matching strategies ensure a sufficient number of points.

Still, the developed descriptors have some advantages over ORB. Both, dBRIEF and mdBRIEF take only 16byte of storage compared to ORB which needs 32byte per descriptor. At the same time, the tracking accuracy (Table 7.1, Table 7.2) and mean percent successfully tracked frames is equally good or better (Table 7.3). At the same time, the additional computational cost is low enough to let the system operate in real-time. The median timings (Table 7.4) for the mapping thread are slightly higher, as more points are matched and thus triangulated and adjusted.

In addition, we analyzed the distribution of tracked points per frame. This is depicted in Figure 7.9 for the two trajectories *Laser 1* and *Laser 2*. The number of tracked points for dBRIEF and mdBRIEF is higher for most frames. For trajectory *Laser 2* it is obvious that mdBRIEF performs best. MultiCol-SLAM with ORB features stopped tracking around frame 350 and is not able to recover, as the MCS moves only in one direction.

7.3.3 Model-Supported Tracking

Finally, the model-supported tracking addition to MultiCol-SLAM presented in Section 7.2.4 is evaluated. This is only possible for the trajectories *Laser 1* and *Laser 2*, as a building model is available. To assess the performance and accuracies of our method, the initial MCS pose $\mathbf{M}_1$ is estimated from model points. Subsequently, we perform the search for image edges described in

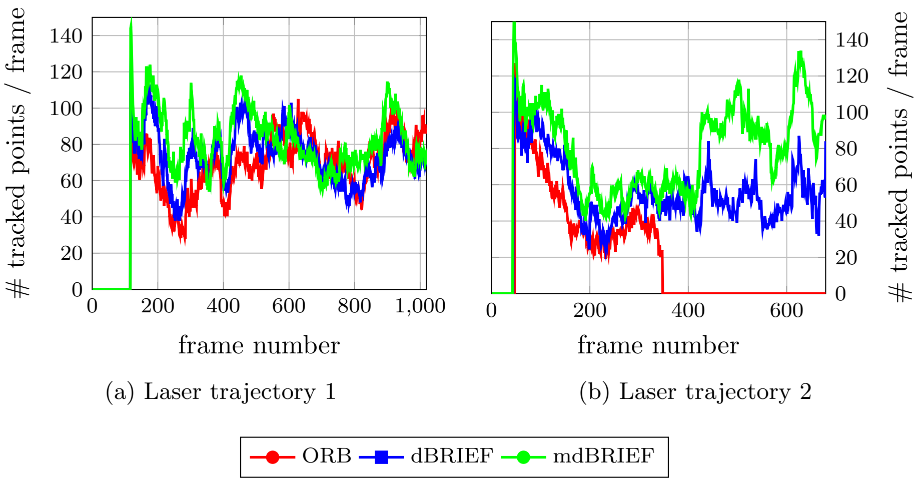


Figure 7.9. Depicted is the number of tracked points for each frame for trajectories *Laser 1* and *Laser 2*.

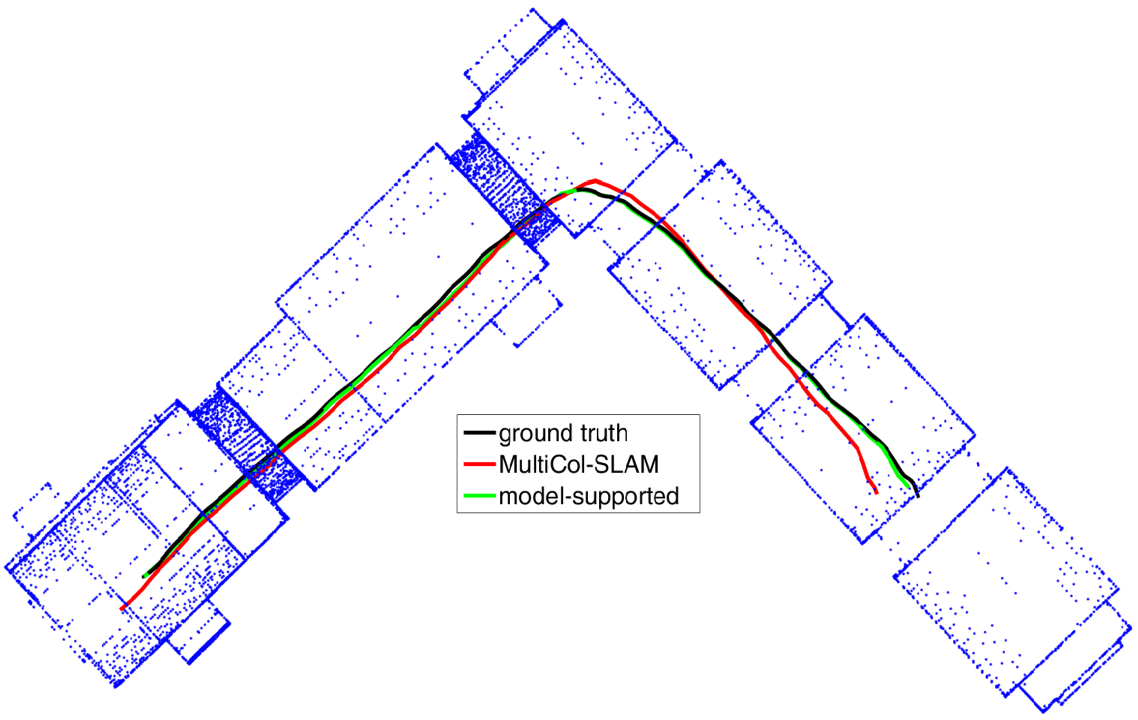


Figure 7.10. Depicts three different trajectories transformed to the building model. The black trajectory is the ground-truth acquired by the lasertracker. The red trajectory represents the MultiCol-SLAM trajectory transformed to the ground truth trajectory by 7DOF alignment. The green trajectory is the trajectory estimated by the model-supported tracking method. During the fast 90° right turn, the SLAM trajectory starts to drift. This drift is avoided by tracking the model edges.

Section 7.2.4 and optimize $\mathbf{M}_1$. In addition, we re-calibrate the MCS by refining all transformation matrices $\mathbf{M}_c$ using the model edges.

The trajectory accuracies are depicted in Table 7.5. It is especially interesting to compare these values to the SLAM only evaluation of the trajectories presented in Table 7.1 and Table 7.2. From these results, it can be assumed that the building model has inferior accuracy compared to what the MCSs pose estimates are capable. Indeed, the model-supported tracking RPE results are slightly inferior. The ATE for the second trajectory however is over two times better. The reason for this is, that the MCS makes a fast right turn at about half of the trajectory. Here, the 'SLAM only' version starts to drift significantly and the scale of the reconstruction suffers. As the trajectory contains no loops, the drift can not be corrected by propagating a loop closing error along the trajectory. This is not the case for the model-supported trajectory estimate, as we always get robust MKF poses that are registered to model having an absolute frame. This result is also qualitatively depicted in Figure 7.10.

7.4 Summary

In this chapter, a multi-fisheye camera SLAM system, called MultiCol-SLAM was proposed. First, we recapitulated the current state-of-the-art in the field and argued why keyframe-based approaches outperform filter-based SLAM systems. In addition, the underground construction site usage scenario was introduced. Arising from the fact that large building models for the construction sites will be available, model-based tracking approaches were subsumed and analyzed leading to open challenges for the fusion of model and point-based approaches and our contributions. Subsequently, we elaborately detailed our framework that builds upon ORB-SLAM and is divided into several threads running in parallel. Then, the building model integration as well as the loop closing mechanisms were explained. Finally, all proposed modules, i.e. multi-camera system, distorted features and model-supported tracking and their impact on the estimated trajectory were examined using accurate ground-truth data.

8. Conclusion and Outlook

In this final chapter, the results and contributions of this thesis are summarized and recapitulated. Finally, each topic is analyzed and potential future work is pointed out.

8.1 Summary and Conclusion

This thesis thoroughly addressed the topic of initial and continuous real-time pose estimation for arbitrary, mobile multi-camera systems in building models. It covers all important aspects ranging from camera calibration to modeling MCSs, feature extraction and description for distorted images to self-localization in building models. Each chapter constitutes a building block that, once combined, lead to the final multi-camera tracking and mapping system. Ultimately, all methods are compiled into a multi-camera SLAM framework and evaluated on accurate ground-truth data. To the best of my knowledge, it is one of the first systems integrating arbitrary MCSs into a full SLAM pipeline, i.e. initialization, tracking, mapping, re-localization, loop closing that is additionally supported by model-based edge tracking. Still, the generality of the developed methods and algorithms was of particular interest, to make them adaptable to a wide range of applications.

In the first chapter, an improved calibration procedure for wide-angle, fisheye and omnidirectional cameras was presented that yields accurate world to image mapping relations for central cameras. By exchanging the residual function of the original procedure, the number of necessary calibration steps was reduced from five to three and the the mean squared reprojection error decreased by a factor of about two.

This improved calibration procedure along with the general camera model provided the foundation for the formulation of MultiCol. This model expands the classic collinearity equation by integrating body to camera frame transformations and the general camera model. In addition, MultiCol is formulated in terms of a hyper graph, alleviating bundle adjustment, camera self- or MCS calibration. The method is evaluated against other approaches and shows state-of-the-art performance w.r.t speed and accuracy.

After having established and evaluated the basic geometric relations, a fast, on-line adaptable, binary descriptor for distorted images was developed. This descriptor adapts to the local distortion of a calibrated camera and in addition learns a binary mask on-line, that suppresses descriptor entries that are not robust against geometric deformations. Coupled with a fast feature detector, the method yields state-of-the-art performance whilst being two times smaller than comparable binary descriptors.

Estimating the initial pose of a MCS w.r.t a textureless building model is a topic that is rarely addressed in literature as this is a very challenging problem. To create synthetic but geometrically correct views of the building model, modern computer graphics techniques were exploited. Then, multiple methods were proposed and evaluated to find the initial camera pose from a large number of pre-rendered model views, leading to a real-time capable self-localization method for MCSs.

Combining the knowledge and methodology gathered and developed throughout the thesis leads to the final model-supported, multi-fisheye camera tracking and mapping framework. First, the difference between filter- and keyframe based methods were analyzed, implicating the superior performance of keyframe based approaches. Thus, for the basic tracking and mapping challenges of our intended system we extended ORB-SLAM, the state-of-the-art in monocular SLAM at the time of writing. To make it applicable to our problem, we integrated the general camera model, MultiCol and the distorted binary features. In addition, we introduced the concept of multi-keyframes to handle multi-cameras at every keyframe pose. Finally, we integrated the spatial information that is available in the form of building models into the tracking and mapping process. By adjusting MKF image edges to building model edges, the system avoids the accumulation of pose errors and thus does not drift. The developed MultiCol-SLAM was evaluated on accurate ground-truth data and showed state-of-the-art performance.

Looking back at the open challenges identified in the introduction concerning egomotion estimation for arbitrary multi-camera systems, it can be concluded that each challenge was addressed in-depth. In most chapters, state-of-the-art results were achieved whilst introducing new contributions. Everything was evaluated on either publicly available data and/or accurate ground-truth. The self-localization chapter set foot on rather new ground and yielded promising results that might lead to exciting future work involving deep learning.

8.2 Outlook

Although the topic of multi-camera egomotion estimation was covered quite exhaustively in this thesis, the presented work opens up a number of interesting research questions and great potential for future work. Thus in this last section, we will pose some ideas.

In Chapter 3 we proposed an improved procedure to calibrate a general camera model. The generality, however, is only valid for central cameras fulfilling the single viewpoint constrain, i.e. all rays are passing through a common projection center. In addition, fisheye cameras are only approximately central as, due to the construction of the lens system, a common projection center can hardly be ensured. Usually, the issue with non-central camera models is basically the increased runtime. This makes them unsuitable for real-time applications and thus for almost any application in this chapter. First work [166] proposes a quasi-central camera model. They show fast runtime and state-of-the-art accuracies for catadioptric cameras. It could be interesting to test, evaluate and potentially adapt this model for the use in MultiCol, the fisheye renderer and the distorted descriptors.

Latter also leaves room for future work. First of all, the distorted descriptor computations are limited to points that have an incident angles smaller than 90° from the principal axis as we undistort and thus project the feature point to a plane. In addition, the descriptors forfeit distinctiveness towards the image borders due to rounding effects of the test positions and increased test correlation. It would be worthwhile to analyze these effects and find possible solutions to alleviate them. Interesting ideas in this direction are proposed by [217]. Learning the test pattern in this thesis was inspired by ORB [154]. We included some studies towards using different detectors for patch extraction. Going a step further, [190] show great improvements changing the unsupervised learning of ORB to their supervised boosting scheme. The need for binary descriptors is particularly driven by the fact that, still nowadays, real-time systems based on feature points are only possible using fast binary features. A more disruptive approach would be to ban the idea of constructing the descriptors by simple binary tests and rather learn the patch to binary descriptor mapping using convolutional neural networks. Latter show great improvements for floating point descriptors [12, 101]. The inference and thus the descriptor creation time of these networks, however, is yet to slow for mobile, real-time applications. However, the progress in the development of small dedicated hardware that is tailored to perform fast inference using CNNs already commenced and it is thus only a matter of time until it becomes feasible. First work on small, fast descriptor networks is presented in [12]. Learning the complete pipeline from detection, description and matching is presented in [212].

Convolutional neural networks could also play an interesting role for MCS self-localization. Instead of extracting and matching edge orientations using hand crafted filters, a siamese network could be learned that transforms real and synthetic images to distinct intermediate representations. First experiments in this direction show promising results. In addition, the re-localization from a previously created map could be improved from using a Bag-of-Binary-Words recognition database to a CNN that regresses the pose directly between MKF poses. First work in this direction is presented in [83] called PoseNet.

Concerning the developed MultiCol-SLAM system, a number of extensions and features come to mind. Direct methods as presented in the state-of-the-art section of Chapter 7 reconstruct the environment with more structure and thus information for further applications. The authors of ORB-SLAM already proposed a semi-dense extension to their work [137] which could be adapted and extended for our MCS SLAM system. To enrich the reconstruction, the combination of our helmet system with a mobile laserscanner is current work-in-progess and first experiments show the feasibility of the approach. Here, the coupling and combination of asynchronous sensor measurements and the on-line MCS to laserscanner calibration are interesting fields of future work. In very recent work [148], this topic is addressed and studied. In addition, the seamless and complementary integration of both sensors into the model-supported MultiCol-SLAM framework for mobile mapping tasks is quite promising. Finally, although CNNs were originally developed for classification, first work shows surprisingly well results in Visual Odometry [35] (frame to frame motion estimation), 3D multi-view reconstruction [31], stereo depth estimation [214] and pose regression [83]. Very recently, gvnn [67] was proposed. Based on STR (Spatial TRansformer) Networks [79], that learn 2D spatial transformations on input patches, the authors propose new types of network layers, that chain projections like world to image mappings, 3D transformations like rotations and translations and so on. So the open question is: will it be possible to learn a (multi-camera) SLAM framework end-to-end?

Bibliography

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building Rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] S. Agarwal and K. Mierle. Ceres Solver: Tutorial & Reference. *Google Inc*, 4, 2012.

[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 72–79. IEEE, 2009.

[4] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast Retina Keypoint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–517. Ieee, 2012.

[5] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. KAZE Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 214–227. Springer, 2012.

[6] P. F. Alcantarilla and T. Solutions. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(7):1281–1298, 2011.

[7] C. Arth, M. Klopschitz, G. Reitmayr, and D. Schmalstieg. Real-Time Self-Localization from Panoramic Images on Mobile Devices. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 37–46. IEEE, 2011.

[8] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit. Instant Outdoor Localization and SLAM Initialization from 2.5 D Maps. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1309–1318, 2015.

[9] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[10] P. Azad, D. Münch, T. Asfour, and R. Dillmann. 6-DoF Model-Based Tracking of Arbitrarily Shaped 3D Objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5204–5209. IEEE, 2011.

[11] P. Baker and Y. Aloimonos. Complete Calibration of a Multi-Camera Network. In *IEEE Workshop on Omnidirectional Vision*, pages 134–141. IEEE, 2000.

[12] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk. PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors. *arXiv preprint arXiv:1601.05030*, 2016.

[13] V. Balntas, L. Tang, and K. Mikolajczyk. BOLD-Binary Online Learned Descriptor For Efficient Image Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2367–2375, 2015.

[14] J. Barreto and K. Daniilidis. Wide Area Multiple Camera Calibration and Estimation of Radial Distortion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2004.

[15] J. P. Barreto and K. Daniilidis. Fundamental Matrix for Cameras With Radial Distortion. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, pages 625–632. IEEE, 2005.

[16] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, 2006.

[17] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez. The Málaga Urban Dataset: High-Rate Stereo and LiDAR in a Realistic Urban Scenario. *International Journal of Robotics Research (IJRR)*, 33(2):207–214, 2014.

[18] J. Bouguet. Camera calibration toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, visited february 2014.

[19] I. N. Bronstein, J. Hromkovic, B. Luderer, H.-R. Schwarz, J. Blath, A. Schied, S. Dempe, G. Wanka, S. Gottwald, E. Zeidler, et al. *Taschenbuch der Mathematik*, volume 1. Springer-Verlag, 2012.

[20] D. C. Brown. Decentering Distortion of Lenses. *Photometric Engineering*, 32(3):444–462, 1966.

[21] M. Brückner, F. Bajramovic, and J. Denzler. Intrinsic and Extrinsic Active Self-Calibration of Multi-Camera Systems. *Machine Vision and Applications*, 25(2):389–403, 2014.

[22] T. Bülow. Spherical Diffusion for 3D Surface Smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(12):1650–1654, 2004.

[23] H. Cai, K. Mikolajczyk, and J. Matas. Learning Linear Discriminant Projections for Dimensionality Reduction of Image Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(2):338–352, 2011.

[24] M. Calonder, V. Lepetit, P. Fua, K. Konolige, J. Bowman, and P. Mihelich. Compact Signatures for High-Speed Interest Point Description and Matching. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 357–364. IEEE, 2009.

[25] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 778–792, 2010.

[26] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (6):679–698, 1986.

[27] G. Carrera, A. Angeli, and A. J. Davison. SLAM-based automatic extrinsic calibration of a multi-camera rig. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2652–2659. IEEE, 2011.

[28] D. Caruso, J. Engel, and D. Cremers. Large-Scale Direct SLAM for Omnidirectional Cameras. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, sep 2015.

[29] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei. Rawseeds Ground Truth Collection Systems for Indoor Self-Localization and Mapping. *Autonomous Robots*, 27(4):353–371, 2009.

[30] C. Choi and H. I. Christensen. Robust 3D visual tracking using particle filtering on the special Euclidean group: A combined approach of keypoint and edge features. *International Journal of Robotics Research (IJRR)*, 31(4):498–519, 2012.

[31] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-r2n2:3D Unified Approach for Single and Multi-View 3D Object Reconstruction. *arXiv preprint arXiv:1604.00449*, 2016.

[32] O. Chum and J. Matas. Matching with PROSAC-Progressive Sample Consensus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 220–226. IEEE, 2005.

[33] B. Clipp, J.-H. Kim, J.-M. Frahm, M. Pollefeys, and R. Hartley. Robust 6DoF Motion Estimation for Non-Overlapping, Multi-Camera Systems. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1–8. IEEE, 2008.

[34] A. Concha and J. Civera. DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 5686–5693. IEEE, 2015.

[35] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia. Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation. *IEEE Robotics and Automation Letters*, 1(1):18–25, 2016.

[36] M. Cummins and P. Newman. Invited Applications Paper FAB-MAP: Appearance-Based Place Recognition and Mapping using a Learned Visual Vocabulary Model. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

[37] K. Daniilidis, A. Makadia, and T. Bülow. Image Processing in Catadioptric Planes: Spatiotemporal Derivatives and Optical Flow Computation. In *Third Workshop on Omnidirectional Vision*, pages 3–10. IEEE, 2002.

[38] Davide Scaramuzza. OCamCalib: Omnidirectional camera calibration toolbox for matlab. https://sites.google.com/site/scarabotix/ocamcalib-toolbox, visited in february 2014, 2014.

[39] A. J. Davison, Y. G. Cid, and N. Kita. Real-Time 3D SLAM With Wide-Angle Vision. In *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[40] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007.

[41] F. Dellaert and M. Kaess. Square Root SAM: Simultaneous Localization and Mapping Via Square Root Information Smoothing. *International Journal of Robotics Research (IJRR)*, 25(12):1181–1203, 2006.

[42] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[43] C. B. Duane. Close-range camera calibration. *Photogrammetric Engineering*, 37:855–866, 1971.

[44] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 834–849. Springer, 2014.

[45] J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015.

[46] C. Engels, H. Stewénius, and D. Nistér. Bundle Adjustment Rules. *Photogrammetric computer vision*, 2:124–131, 2006.

[47] A. Ess, A. Neubeck, and L. J. Van Gool. Generalised Linear Pose Estimation. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–10, 2007.

[48] M. Everingham, A. Zisserman, C. K. Williams, and L. Van Gool. The PASCAL visual object classes challenge 2006 (VOC2006) results, 2006.

[49] B. Fan, Z. Wang, and F. Wu. Local Image Descriptor: Modern Approaches, 2016.

[50] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 321–334. Springer, 1992.

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

[52] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. IEEE, 2014.

[53] W. Förstner. Minimal Representations for Uncertainty and Estimation in Projective Spaces. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 619–632. Springer, 2010.

[54] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle Filters for Mobile Robot Localization. In *Sequential Monte Carlo methods in practice*, pages 401–428. Springer, 2001.

[55] J.-M. Frahm, K. Köser, and R. Koch. Pose Estimation for Multi-Camera Systems. In *Pattern Recognition*, pages 286–293. Springer, 2004.

[56] O. Frank, R. Katz, C.-L. Tisse, and H. F. Durrant-Whyte. Camera Calibration for Miniature, Low-cost, Wide-angle Imaging Systems. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–10. Citeseer, 2007.

[57] A. Furnari, G. M. Farinella, A. R. Bruna, and S. Battiato. Generalized Sobel filters for gradient estimation of distorted images. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 3250–3254. IEEE, 2015.

[58] J. Gallier. *Geometric Methods and Applications: for Computer Science and Engineering*, volume 38. Springer Science & Business Media, 2011.

[59] D. Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[60] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete Solution Classification for the Perspective-Three-Point Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):930–943, 2003.

[61] E. Garcia-Fidalgo and A. Ortiz. Vision-based topological mapping and localization methods: A survey. *Robotics and Autonomous Systems*, 64:1–20, 2015.

[62] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision (IJCV)*, 94(3):335–360, 2011.

[63] A. Geiger, P. Lenz, and R. Urtasun. Are we Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012.

[64] C. Geyer and K. Daniilidis. A Unifying Theory for Central Panoramic Systems and Practical Implications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 445–461. Springer, 2000.

[65] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle Filters for Positioning, Navigation, and Tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, 2002.

[66] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3279–3286, 2015.

[67] A. Handa, M. Bloesch, V. Patraucean, S. Stent, J. McCormac, and A. Davison. gvnn: Neural Network Library for Geometric Computer Vision. *arXiv preprint arXiv:1607.07405*, 2016.

[68] P. Hansen, W. Boles, and P. Corke. Spherical Diffusion for Scale-Invariant Keypoint Detection in Wide-Angle Images. In *Proceedings of the Digital Image Computing on Techniques and Applications (DICTA)*, pages 525–532. IEEE, 2008.

[69] A. Harmat, I. Sharf, and M. Trentini. Parallel Tracking and Mapping with Multiple Cameras on an Unmanned Aerial Vehicle. In *Intelligent robotics and applications*, pages 421–432. Springer, 2012.

[70] A. Harmat, M. Trentini, and I. Sharf. Multi-Camera Tracking and Mapping for Unmanned Aerial Vehicles in Unstructured Environments. *Journal of Intelligent & Robotic Systems*, 78(2):291–317, 2015.

[71] C. Harris and C. Stennett. RAPID-a video rate object tracker. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–6, 1990.

[72] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[73] R. I. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. University Press, Cambridge, UK, 2008.

[74] J. Heinly, E. Dunn, and J.-M. Frahm. Comparative evaluation of binary features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 759–773. Springer, 2012.

[75] L. Heng, G. H. Lee, and M. Pollefeys. Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle. In *Proceedings of Robotics: Science and Systems (RSS)*, 2014.

[76] J. A. Hesch and S. I. Roumeliotis. A Direct Least-Squares (DLS) Method for PnP. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 383–390. IEEE, 2011.

[77] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(5):876–888, 2012.

[78] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-Form Solution of Absolute Orientation using Orthonormal Matrices. *Journal of the Optical Society of America*, 5:1127–1136, 1988.

[79] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[80] F. Jurie and B. Triggs. Creating Efficient Codebooks for Visual Recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, pages 604–610. IEEE, 2005.

[81] J. Kannala and S. S. Brandt. A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(8):1335–1340, 2006.

[82] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–506. IEEE, 2004.

[83] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2015.

[84] B. Kenwright. A Beginners Guide to Dual-Quaternions. *WSCG'2012*, 2012.

[85] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 2100–2106. IEEE, 2013.

[86] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 225–234. IEEE, 2007.

[87] G. Klein and D. Murray. Improving the agility of keyframe-based SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 802–815. Springer, 2008.

[88] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–86. IEEE, 2009.

[89] G. Klein and D. W. Murray. Full-3D Edge Tracking with a Particle Filter. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1119–1128, 2006.

[90] L. Kneip and P. Furgale. OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2014.

[91] L. Kneip and P. Furgale. OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2014.

[92] L. Kneip, P. Furgale, and R. Siegwart. Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPnP Problem. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3770–3776. IEEE, 2013.

[93] L. Kneip and H. Li. Efficient Computation of Relative Pose for Multi-Camera Systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 446–453, 2014.

[94] L. Kneip, H. Li, and Y. Seo. UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 127–142. Springer, 2014.

[95] L. Kneip, D. Scaramuzza, and R. Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2969–2976. IEEE, 2011.

[96] L. Kneip, R. Siegwart, and M. Pollefeys. *Finding the exact rotation between two images independently of the translation*. Springer, 2012.

[97] K. Kraus. *Photogrammetrie: Geometrische Informationen aus Photographien und Laserscanneraufnahmen*. Walter de Gruyter, 2004.

[98] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother. Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 954–962, 2015.

[99] Z. Kukelova, J. Heller, M. Bujnak, and T. Pajdla. Radial distortion homography. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 639–647, 2015.

[100] R. K. Kumar, A. Ilie, J.-M. Frahm, and M. Pollefeys. Simple Calibration of Non-Overlapping Cameras with a Mirror. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7. IEEE, 2008.

[101] V. Kumar B G, G. Carneiro, and I. Reid. Learning Local Image Descriptors With Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[102] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G^2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613. IEEE, 2011.

[103] G. Kurillo, Z. Li, and R. Bajcsy. Wide-Area External Multi-Camera Calibration using Vision Graphs and Virtual Calibration Object. In *Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–9. IEEE, 2008.

[104] H. Lategahn, J. Beck, B. Kitt, and C. Stiller. How to learn an illumination robust image feature for place recognition. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 285–291. IEEE, 2013.

[105] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision (IJCV)*, 81(2):155–166, 2009.

[106] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE, 2011.

[107] G. Levi and T. Hassner. LATCH: Learned Arrangements of Three Patch Codes. *arXiv preprint arXiv:1501.03719*, 2015.

[108] B. Li, L. Heng, K. Koser, and M. Pollefeys. A Multiple-Camera System Calibration Toolbox using a Feature Descriptor-based Calibration Pattern. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1301–1307. IEEE, 2013.

[109] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide Pose Estimation using 3D Point Clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 15–29. Springer, 2012.

[110] H. Lim, J. Lim, and H. J. Kim. Real-Time 6-DoF Monocular Visual SLAM in a Large-Scale Environment. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1532–1539. IEEE, 2014.

[111] K.-H. Lin, C.-H. Chang, A. Dopfer, and C.-C. Wang. Mapping and Localization in 3D Environments Using a 2D Laser Scanner and a Stereo Camera. *Journal of information science and engineering*, 28(1):131–144, 2012.

[112] M. A. Lourakis and A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):1–30, 2009.

[113] M. I. Lourakis and A. A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2, 2009.

[114] M. Lourenço, J. P. Barreto, and F. Vasconcelos. sRD-SIFT: Keypoint Detection and Matching in Images with Radial Distortion. *IEEE Transactions on Robotics*, 28(3):752–760, 2012.

[115] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[116] T. Luhmann, S. Robson, S. Kyle, and I. Harley. *Close Range Photogrammetry: Principles, Methods and Applications.* Whittles, 2006.

[117] D. S. Ly, C. Demonceaux, P. Vasseur, and C. Pégard. Extrinsic Calibration of Heterogeneous Cameras by Line Images. *Machine Vision and Applications*, 25(6):1601–1614, 2014.

[118] H.-G. Maas. Image Sequence Based Automatic Multi-Camera System Calibration Techniques. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(5):352–359, 1999.

[119] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and Generic Corner Detection Based on the Accelerated Segment Test. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2010.

[120] A. L. Majdik, Y. Albers-Schoenberg, and D. Scaramuzza. MAV Urban Localization from Google Street View Data. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3979–3986. IEEE, 2013.

[121] C. Mei. Omnidirectional Calibration Toolbox. http://www.robots.ox.ac.uk/˜cmei/Toolbox.html, visited in february 2014, 2014.

[122] C. Mei and P. Rives. Single View Point Omnidirectional Camera Calibration from Planar Grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, apr 2007.

[123] B. Micusık. *Two-view Geometry of Omnidirectional Cameras.* PhD thesis, Czech Technical University, 2004.

[124] B. Micušık and T. Pajdla. Structure from Motion With Wide Circular Field of View Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(7):1135–1149, 2006.

[125] B. Micusik and H. Wildenauer. Descriptor Free Visual Indoor Localization with Line Segments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3165–3173, 2015.

[126] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-DoF Localization on Mobile Devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–283. Springer, 2014.

[127] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(10):1615–1630, oct 2005.

[128] O. Miksik and K. Mikolajczyk. Evaluation of Local Detectors and Descriptors for Fast Feature Matching. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 2681–2684. IEEE, 2012.

[129] M. Montemerlo and S. Thrun. FastSLAM 2.0. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pages 63–90, 2007.

[130] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *AAAI/IAAI*, pages 593–598, 2002.

[131] J.-M. Morel and G. Yu. ASIFT: A new Framework for Fully Affine Invariant Image Comparison. *Journal on Imaging Sciences (SIAM)*, 2(2):438–469, 2009.

[132] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. BLORT-the Blocks World Robotic Vision Toolbox. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[133] D. Muhle, S. Abraham, C. Heipke, and M. Wiggenhagen. Estimating the Mutual Orientation in a Multi-Camera System with a Non Overlapping Field of View. In *Photogrammetric Image Analysis (PIA)*, pages 13–24. Springer, 2011.

[134] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *Proceedings of the Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, pages 331–340. INSTICC Press, 2009.

[135] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[136] R. Mur-Artal and J. D. Tardos. ORB-SLAM: tracking and mapping recognizable features. In *Proceedings of Robotics: Science and Systems (RSS)*, 2014.

[137] R. Mur-Artal and J. D. Tardós. Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM. *Proceedings of Robotics: Science and Systems, Rome, Italy*, 2015.

[138] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE, 2011.

[139] K. Nickel and R. Stiefelhagen. Dynamic Integration of Generalized Cues for Person Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–526. Springer, 2008.

[140] W. Niemeier. *Ausgleichungsrechnung: Statistische Auswertemethoden.* Walter de Gruyter, 2008.

[141] K. Pirker, M. Rüther, and H. Bischof. Histogram of Oriented Cameras-A New Descriptor for Visual SLAM in Dynamic Environments. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12, 2010.

[142] K. Pirker, M. Rüther, and H. Bischof. CD SLAM-Continuous Localization and Mapping in a Dynamic World. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3990–3997. IEEE, 2011.

[143] S. J. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.

[144] L. Puig, Y. Bastanlar, P. Sturm, J. J. Guerrero, and J. Barreto. Calibration of Central Catadioptric Cameras Using a DLT-like Approach. *International Journal of Computer Vision (IJCV)*, 93(1):101–114, 2011.

[145] L. Puig, J. Bermúdez, P. Sturm, and J. Guerrero. Calibration of Omnidirectional Cameras in Practice: A Comparison of Methods. *Computer Vision and Image Understanding*, 116(1):120 – 137, 2012.

[146] L. Puig and J. J. Guerrero. Scale Space for Central Catadioptric Systems: Towards a Generic Camera Feature Extractor. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1599–1606. IEEE, 2011.

[147] D. A. Randall, T. D. Ringler, R. P. Heikes, P. Jones, J. Baumgardner, et al. Climate Modeling with Spherical Geodesic Grids. *Computing in Science and Engineering*, 4(5):32–41, 2002.

[148] J. Rehder, R. Siegwart, and P. Furgale. A General Approach to Spatiotemporal Calibration in Multisensor Systems. *IEEE Transactions on Robotics (T-RO)*, 32(2):383–398, 2016.

[149] G. Reitmayr and T. Drummond. Going Out: Robust Model-based Tracking for Outdoor Augmented Reality. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2006.

[150] A. Richardson and E. Olson. TailoredBRIEF: Online Per-Feature Descriptor Customization. *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015.

[151] E. Rosten and T. Drummond. Fusing Points and Lines for High Performance Tracking. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1508–1515, Peking, oct 2005.

[152] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 430–443. Springer, 2006.

[153] E. Rosten, R. Porter, and T. Drummond. Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(1):105–119, 2010.

[154] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011.

[155] M. Rufli, D. Scaramuzza, and R. Siegwart. Automatic Detection of Checkerboards on Vlurred and Distorted Images. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3121–3126. IEEE, 2008.

[156] R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. Davison. Slam++: Simultaneous Localisation and Mapping at the Level of Objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359, 2013.

[157] T. Sattler, B. Leibe, and L. Kobbelt. Improving Image-Based Localization by Active Correspondence Search. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 752–765, 2012.

[158] D. Scaramuzza, A. Martinelli, and R. Siegwart. A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion. In *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems (ICVS*, pages 45–45. IEEE, 2006.

[159] D. Scaramuzza, A. Martinelli, and R. Siegwart. A Toolbox for Easily Calibrating Omnidirectional Cameras. In *Proceedings of the Annual Conference of the Robotics Society of Japan (RSJ)*, pages 5695–5701. IEEE, 2006.

[160] G. Schindler, M. Brown, and R. Szeliski. City-Scale Location Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7. IEEE, 2007.

[161] J. Schmidt and H. Niemann. Using Quaternions for Parametrizing 3-D Rotations in Unconstrained Nonlinear Optimization. In *VMV*, volume 1, pages 399–406, 2001.

[162] D. Schneider, E. Schwalbe, and H.-G. Maas. Validation of Geometric Models for Fisheye Lenses. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(3):259–266, 2009.

[163] J. Schneider and W. Förstner. Bundle Adjustment and System Calibration with Points at Infinity for Omnidirectional Camera Systems. *Z. f. Photogrammetrie, Fernerkundung und Geoinformation*, 4:309–321, 2013.

[164] J. Schneider, F. Schindler, T. Läbe, and W. Förstner. Bundle Adjustment for Multi-Camera Systems with Points at Infinity. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:75–80, 2012.

[165] M. Schönbein and A. Geiger. Omnidirectional 3D Reconstruction in Augmented Manhattan Worlds. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2014.

[166] M. Schönbein, T. Straus, and A. Geiger. Calibrating and Centering Quasi-Central Catadioptric Cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4443–4450. IEEE, 2014.

[167] T. Schöps, Thomas, J. Engel, and D. Cremers. Semi-Dense Visual Odometry for AR on a Smartphone. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 145–150. IEEE, 2014.

[168] G. Schweighofer and A. Pinz. Globally Optimal O(n) Solution to the PnP Problem for General Camera Models. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–10, 2008.

[169] J. Shi and C. Tomasi. Good Features to Track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600. IEEE, 1994.

[170] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 118–126, 2015.

[171] N. Snavely, S. M. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846, 2006.

[172] C. Steger. Occlusion, Clutter, and Illumination Invariant Object Recognition. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A):345–350, 2002.

[173] H. Stewenius, C. Engels, and D. Nistér. Recent Developments on Direct Relative Orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.

[174] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to Minimal Generalized Relative Pose Problems. In *Workshop on Omnidirectional Vision*, volume 1, page 3, 2005.

[175] H. Strasdat. *Local Accuracy and Global Consistency for Efficient Visual Slam.* PhD thesis, Citeseer, 2012.

[176] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolig. Double Window Optimisation for Constant Time Visual SLAM. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2352–2359. IEEE, 2011.

[177] H. Strasdat, J. Montiel, and A. J. Davison. Scale Drift-Aware Large Scale Monocular SLAM. In *Robotics: Science and Systems*, volume 2, page 5, 2010.

[178] H. Strasdat, J. M. Montiel, and A. J. Davison. Visual SLAM: Why Filter? *Image and Vision Computing*, 30:65–77, 2012.

[179] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved Matching with Smaller Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 34(1):66–78, 2012.

[180] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 573–580. IEEE, 2012.

[181] T. Svoboda. Quick Guide to Multi-Camera Self-Calibration. *ETH, Swiss Federal Institute of Technology, Zurich, Tech. Rep. BiWi-TR-263*, 2003.

[182] C. Sweeney. *Theia Multiview Geometry Library: Tutorial & Reference.* University of California Santa Barbara.

[183] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. gDLS: A Scalable Solution to the Generalized Pose and Scale Problem. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 16–31. Springer, 2014.

[184] C. Sweeney, L. Kneip, T. Hollerer, and M. Turk. Computing Similarity Transformations from Only Image Correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3305–3313, 2015.

[185] C. Sweeney, T. Sattler, T. Hollerer, M. Turk, and M. Pollefeys. Optimizing the Viewing Graph for Structure-from-Motion. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 801–809, 2015.

[186] R. Szeliski. *Computer Vision: Algorithms and Applications.* Springer Science & Business Media, 2010.

[187] C. Teuliere, E. Marchand, and L. Eck. 3-D Model-Based Tracking for UAV Indoor Localization. *IEEE Transactions on Cybernetics*, 45(5):869–879, 2015.

[188] E. Tola, V. Lepetit, and P. Fua. A Fast Local Descriptor for Dense Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[189] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle Adjustment-a Modern Synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[190] T. Trzcinski, C. M. Christoudias, and V. Lepetit. Learning Image Descriptors with Boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2015.

[191] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting Binary Keypoint Descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2874–2881. IEEE, 2013.

[192] T. Trzcinski, M. Christoudias, V. Lepetit, and P. Fua. Learning Image Descriptors with the Boosting-Trick. In *Advances in Neural Information Processing Systems*, pages 269–277, 2012.

[193] T. Trzcinski and V. Lepetit. Efficient Discriminative Projections for Compact Binary Descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[194] M. Ulrich, C. Wiedemann, and C. Steger. CAD-Based Recognition of 3D Objects in Monocular Images. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1191–1198, Kobe, Japan, may 2009.

[195] S. Urban, J. Leitloff, and S. Hinz. Improved Wide-Angle, Fisheye and Omnidirectional Camera Calibration. *ISPRS Journal of Photogrammetry and Remote Sensing*, 108:72–79, 2015.

[196] S. Urban, J. Leitloff, and S. Hinz. MLPnP: A Real-Time Maximum Likelihood Solution to the Perspectinve-N-Point problem. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science*, 2016.

[197] S. Urban, J. Leitloff, S. Wursthorn, and S. Hinz. Self-Localization of a Multi-Fisheye Camera Based Augmented Reality System in Textureless 3D Building Models. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science*, 2:43–48, 2013.

[198] S. Urban and M. Weinmann. Finding a Good Feature Detector-Descriptor Combination for the 2d Keypoint-Based Registration of Tls Point Clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:121–128, 2015.

[199] S. Urban, S. Wursthorn, J. Leitloff, and S. Hinz. MultiCol Bundle Adjustment: A Generic Method for Pose Estimation, Simultaneous Self-Calibration and Reconstruction for Arbitrary Multi-Camera Systems. *International Journal of Computer Vision (IJCV)*, pages 1–19, 2016.

[200] L. Vacchetti, V. Lepetit, and P. Fua. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, Arlington, November 2004.

[201] L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking Using Online and Offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(10):1385–1391, 2004.

[202] J. Ventura, C. Arth, and V. Lepetit. An Efficient Minimal Solution for Multi-Camera Motion. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 747–755, 2015.

[203] M. Weinmann. Visual Features - From Early Concepts to Modern Computer Vision. In *Advanced Topics in Computer Vision*, pages 1–34. Springer, 2013.

[204] A. Wendel, M. Maurer, and H. Bischof. Visual Landmark-Based Localization for MAVs Using Incremental Feature Updates. In *DIMPVT*, pages 278–285. IEEE, 2012.

[205] S. Winder, G. Hua, and M. Brown. Picking the Best Daisy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 178–185. IEEE, 2009.

[206] P. Wohlhart and V. Lepetit. Learning Descriptors for Object Recognition and 3D Pose Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3109–3118, 2015.

[207] J. Wolf, W. Burgard, and H. Burkhardt. Robust Vision-Based Localization by Combining an Image-Retrieval System with Monte Carlo Localization. *IEEE Transactions on Robotics*, 21(2):208–216, 2005.

[208] C. Wu. VisualSFM: A visual structure from motion system. *URL: http://homes. cs. washington. edu/˜ ccwu/vsfm*, 9, 2011.

[209] C. Wu. Towards Linear-Time Incremental Structure from Motion. In *2013 International Conference on 3D Vision-3DV*, pages 127–134. IEEE, 2013.

[210] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore Bundle Adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3057–3064. IEEE, 2011.

[211] H. Wuest, F. Wientapper, and D. Stricker. Adaptable Model-Based Tracking Using Analysis-by-Synthesis Techniques. In *Computer analysis of images and patterns (CAIP)*, volume 4673 of *LNCS*, pages 20–27, Wien, aug 2007. Springer Berlin/Heidelberg.

[212] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. *arXiv preprint arXiv:1603.09114*, 2016.

[213] J. Yu and L. McMillan. General Linear Cameras. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 14–27. Springer, 2004.

[214] J. Zbontar and Y. LeCun. Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *Journal of Machine Learning Research*, 17:1–32, 2016.

[215] B. Zeisl, T. Sattler, and M. Pollefeys. Camera pose voting for large-scale image-based localization. In *ICCV*, pages 2704–2712, 2015.

[216] Z. Zhang. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 1, pages 666–673. IEEE, 1999.

[217] Q. Zhao, W. Feng, L. Wan, and J. Zhang. SPHORB: A Fast and Robust Binary Feature on the Sphere. *International Journal of Computer Vision (IJCV)*, 113(2):143–159, 2015.

[218] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisiting the PnP Problem: A Fast, General and Optimal Solution. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2344–2351. IEEE, 2013.

[219] D. Zou and P. Tan. COSLAM: Collaborative Visual Slam in Dynamic Environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(2):354–366, 2013.

A. Acknowledgements

First I would like to thank my doctoral adviser Stefan Hinz for enabling me to do my Ph.D. at his institute. If he wouldn't have asked me in the first place, I would probably not have considered to pursue this path. Furthermore, I would like to thank some people from the IPF in particular:

André Dittrich for bearing me in his (our) office.

Martin Weinmann for helpful tips and tricks about publishing (and supplying me with honey).

Sven Wursthorn and Jens Leitloff for fruitful discussions, IT help and proof-reading.

Boris Jutzi for giving me the chance to teach some courses and the occasional ”Alkoholfreies Weizen” after early Feierabend bike rides.

Markus Hillemann for his friendship, letting me win some squash matches and our regular winter holidays.

This dissertation, however, would not have been possible without the unlimited support of my parents Gerd and Ruth. You have always given me the chance to be independent and free in my decisions and to pursue my personal goals. Last but not least I would like to thank Clémence for proof-reading, simply being there, listening and sharing this chapter of our lives together.