UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Camera-Agnostic Monocular SLAM and Semi-Dense 3D Reconstruction

Masterarbeit
zur Erlangung des Grades
Master of Science
im Studiengang Computervisualistik

vorgelegt von

Martin Rünz

Betreuer: Dipl.-Inform. F. Neuhaus, Dipl.-Inform. C. Winkens, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Zweitgutachter: Dipl.-Inform. F. Neuhaus, Dipl.-Inform. C. Winkens, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im September 2015

Kurzfassung

Die folgende Arbeit behandelt Techniken zur Lokalisierung und Kartierung unter der Verwendung einer einzigen Kamera. Das zugrundeliegende Problem, auch als *monocular SLAM* bekannt, wird zunächst erörtert und im Kontext bestehender Publikationen betrachtet. Es werden relevante mathematische Konzepte präsentiert, um nachfolgend diskutierte Verfahren untersuchen zu können. Dabei wird ausführlich auf die Methodik von Verfahren eingegangen, welche dem Stand der Technik entsprechen.
Ein besonderer Fokus der Arbeit besteht darin, neben traditionellen auch omnidirektionale Kameras miteinzubeziehen. Durch die Funktionsweise omnidirektionaler Kameras ergeben sich besondere Anforderungen, denen übliche SLAM-Verfahren nicht entsprechen. Daher werden explizit jene Methoden untersucht, die sowohl für den Einsatz mit traditionellen als auch mit omnidirektionalen Kameras geeignet sind. Abschließend wird ein neu entwickeltes Verfahren vorgestellt, das sich am Stand der Technik orientiert und durch Verallgemeinerungen mit beliebigen zentralen Kameramodellen einsetzbar ist. Die Evaluation des neuen, CAM-SLAM genannten Verfahrens zeigt, dass dieses ähnlich genau wie modernste Referenzverfahren operiert, gleichzeitig aber eine höhere Flexibilität bietet.

Abstract

The following thesis discusses localization and mapping techniques based on a single camera. After introducing the given problem, which is known as *monocular SLAM*, an overview of related publications is provided. Relevant mathematical principles are presented and subsequently used to compare available methods in the abstract. During this comparison, state-of-the-art methods are analyzed thoroughly.
Various camera models are studied with emphasis on omnidirectional cameras, and corresponding techniques are investigated. Employing omnidirectional cameras imposes special requirements that are not met by common SLAM-methods. In this thesis, techniques that are applicable for traditional as well as omnidirectional cameras are evaluated.
A new **c**amera **a**gnostic **m**onocular SLAM system (CAM-SLAM) is presented. It was developed within the scope of this thesis and is inspired by recently proposed SLAM-methods. In contrast to most other systems, it supports any central camera model. Experiments show that CAM-SLAM features similar accuracy as state-of-the-art methods, while being considerably more flexible.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.
Die Vereinbarung der Arbeitsgruppe für Studien- und Abschlussarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☒ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☒ nein ☐

Koblenz, den 30. September 2015

Contents

Chapter 1

Introduction

The term *simultaneous localization and mapping* (*SLAM*) denotes the process of creating or refining a map, while determining the own position at the same time. This is a trivial task for humans: Imagine waking up after a long drive as a passenger, facing a new environment. Usually, it will only take seconds to get a proper understanding of the surroundings. Designing a machine with similar perceptional capabilities is a complex goal, which researchers of different domains strive for by building sensors and software. Inspired by the effective vision systems one can find in nature, such as the human eye in conjunction with the visual cortex, researchers have brought computer vision into focus.

SLAM techniques that rely on vision are referred to as *visual-SLAM* techniques and their performance increased remarkably in the last decade. This thesis concentrates on *omnidirectional monocular* visual-SLAM systems, which are SLAM systems that deploy a single camera[1] with a wide field of view. Taking up the analogy to nature, such a vision system would correspond to that of a rabbit or horse, as the eyes of these animals also maximize their field of view at the expense of having monocular vision. There are several reasons for advocating the use of omnidirectional sensors in computer vision. First, many algorithms with applications to robotics detect visual landmarks. These landmarks will be present in many images when an omnidirectional camera is employed, resulting in a higher robustness. Second, a vision system with a high field of view collects more data than a traditional vision system in the same amount of time, although at a lower resolution. This is particularly useful in robotics or when teleconferencing or performing surveillance.

The purpose of this thesis is to introduce concepts of visual-SLAM, and the design of a new SLAM system for omnidirectional cameras. In fact, the newly creat-

[1] Or a set of cameras with non-overlapping images.

ed system makes little assumptions about the camera model and supports traditional, omnidirectional and every single-view camera model[2]. For this reason, it is called ***c**amera-**a**gnostic **m**onocular SLAM* – CAM-SLAM. Besides performing monocular SLAM, CAM-SLAM is also capable of reconstructing the environment semi-densely.

Firstly, the thesis presents related work in a concise and rather historical manner. Relevant publications are revisited later and in more detail. In order to discuss visual-SLAM methods on a conceptional level, a theoretical background is provided in Chapter 2 and additional resources are referenced when necessary. Since CAM-SLAM supports every central camera model, a set of common models is presented in Chapter 3. The models presented in this chapter are also used during the experiments performed later. Chapter 4 continues to elaborate on theoretical aspects related to multi-view geometry. It already aims at camera model-agnostic methods and first tests are run. State-of-the-art monocular visual-SLAM methods are investigated and compared in Chapter 5. Both keypoint-based and direct methods are considered and omnidirectional methods are reviewed as well. The subsequent Chapter 6 analyzes semi-dense 3D reconstruction algorithms and provides the mathematical derivation of the method implemented in CAM-SLAM. As loop-closing is an important aspect of large-scale SLAM-systems, its fundamentals are presented in Chapter 7. The remaining chapters are dedicated to the implementation and evaluation of CAM-SLAM. The combination and modification of algorithms are discussed and the system's performance is measured during experiments.

1.1 Related work

A substantial amount of the theoretic principles involved in omnidirectional machine vision has been established during the 1990s, followed by the implementation of thorough omnidirectional vision systems. The first decisive monocular SLAM systems emerged in the subsequent decade and naturally inspired omnidirectional monocular SLAM systems. A roughly chronological overview of related work follows, while more details on relevant publications are given in the respective chapters of this thesis.

Modern monocular SLAM systems perform a 3D reconstruction of parts of the environment for mapping. Early works on omnidirectional 3D reconstruction were the ones of Ishiguro et al. [IYT92] and Kang and Szeliski [KS97]. While Ishiguro

[2] As long as it is possible to extract and track salient image features.

et al. used a single rotating camera[3] to create a panoramic depth image, Sing Bing Kang and Richard Szeliski showed that the 8-point algorithm can be used to recover structure from multiple panoramic images using normalized cylindrical coordinates.

In 1998, Gluckman and Nayar [GN98] showed that it is possible to estimate the motion of a catadioptric camera using the gradient based optical flow computation of Lucas and Kanade [LK81]. Simon Baker and Shree Nayar also derived further theoretical aspects of catadioptric camera systems in [BN98, BN99]. They describe which configurations exhibit a single effective viewpoint and that the image resolution at the periphery of an omnidirectional image is highest.

A robot that performs omnidirectional visual homing – the task of navigating back to a start location based on computer vision, similarly to insects returning to their nests – was presented by Franz et al. [FSMB98] and later by Argyros et al. [ABO01] and Goedemé et al. [GTVG+05].

In [CS99], Chahl and Srinivasan developed an algorithm to estimate the distance of objects given their image space deformation while moving. Geyer and Daniilidis continued to investigate geometrical characteristics of catadioptric cameras [GD99, GD01] and proposed a unifying camera model [GD00] for traditional and omnidirectional cameras. In order to reduce processing time, Hicks and Bajcsy focused on mirror designs that do not distort the ground plane in their work [HB01].

Bunschoten and Kröse use of a cylindrical camera model to perform a 3D reconstruction in [BK01b, BK03]. They argue that by employing the cylindrical model, the parametrization of epipolar curves is easier than the parametrization of epipolar conics, as described by Tomáš Svoboda in [Svo00b]. A method to compute such cylindrical images based on a catadioptric video stream was earlier proposed by Peri and Nayar [PN97].

One of the most prominent publications on monocular SLAM is the seminal work of Andrew Davison published in 2003 [Dav03]. Davison applied EKF-SLAM to the domain of computer vision and his method is able to execute mapping as well as tracking in real-time. In 2006, Eade and Drummond presented a competing method [ED06] based on Fast-SLAM, which scales better.

Given the motion of a robot, Fleck et al. [FBB+05] were able to reconstruct a dense 3D map of the environment using a catadioptric setup. On that account, they employed a *graph cuts*-based stereo matching algorithm.

Calibration methods for omnidirectional cameras were presented by Scaramuzza et al. [SMS06a] as well as Mei and Rives [MR07] and are based on the model earlier proposed by Geyer and Daniilidis.

[3]The actual optical center of the camera was rotating and translating in a predefined way, allowing the computation of depth values.

Some authors, such as Zhu et al. [ZHK+07], acquire omnidirectional vision with an array of traditional cameras. The drawbacks of these arrays are that they do not exhibit a single viewpoint and that they are expensive, since multiple cameras have to be installed. Goedemé et al. [GNTVG07], Murillo et al. [MGS07] as well as Valgren and Lilienthal [VL07] presented global omnidirectional topological localization systems. All three systems are able to localize themselves, though requiring a pre-built map. A 9-point algorithm for estimating para-catadioptric fundamental matrices was developed by Geyer and Stewenius [GS07]. In this thesis, however, the computation of essential matrices is preferred, as described in Section 4.1.

A different method of computing the essential matrix using a para-catadioptric sensor is given by Gebken and Sommer [GS08]. It is based on conformal geometric algebra and non-linear optimization techniques.

In 2009, the next milestone in monocular SLAM was achieved by Klein and Murray [KM09], through decoupling the mapping and tracking task. As a result, a more expensive mapping method is used, while tracking is performed in real-time.

The concept of homographies is frequently encountered in computer vision. For instance, the map initialization of a SLAM system can be based on homographies. Zhang et al. [ZLZH10] developed a method to compute homographies in the presence of a catadioptric camera with hyperbolic or elliptical mirror.

In 2011, Kawanishi et al. [KYK11] and Pagani and Stricker [PS11] presented a catadioptric and a spherical structure from motion application, respectively. Both systems are not intended for the use in real-time domains but share mathematical similarities, as will be revisited later. Schönbein et al. showed that the quality of three-view omnidirectional 3D reconstruction outperforms two-view reconstruction, when beneficial camera locations are chosen [SRL13].

There has been a trend in recent years to compute visual odometry, i.e. to estimate the robot motion, using omnidirectional vision [Lab06, Sca08, SS08, TPD08, Sch12, ATA13]. The advantage of estimating motion with omnidirectional sensors lies in the computation of the rotational motion component. While rotations are usually a burden when working with traditional cameras, all methods formerly mentioned exploit omnidirectional peculiarities to compute rotation. Although some of the systems, such as the one of Scaramuzza and Siegwart [SS08], exhibit a high accuracy, their estimation will diverge from the ground truth eventually. This happens because small errors accumulate over time, resulting in an inevitable drift. Furthermore, some of the systems impose constraints to the environment, which the system of Scaramuzza and Siegwart is a good example for, again. Such a constraint can be the assumption that the robot is moving on flat ground only, or that the robot is surrounded by walls.

In contrast to visual odometry approaches, researchers have also adapted full-fledged SLAM systems to the domain of omnidirectional vision, often based on EKF-SLAM [RPG10b, RPG10a, GRMG11, PO13] or Fast-SLAM [GMR13]. A detailed discussion on monocular SLAM will follow in Section 5.

Please note that the very recent omnidirectional implementation of LSD-SLAM [CEC15] was presented too late to be investigated extensively. It is referred to for the sake of completeness and is only addressed shortly in the remaining.

1.2 Hardware

(a) An omnidirectional camera built by attaching a hyperboloid mirror to a DSLR camera.

(b) Omnidirectional image made with the camera setup shown on the left.

Figure 1.1: Prototypical omnidirectional camera and a corresponding image. Courtesy of Ryosuke Kawanishi and Toru Kaneko, Shizuoka University and Atsushi Yamashita and Hajime Asama, University of Tokyo. Published in [KYKA12] as Figure 1.

As it is denoted in [BK01a, p.4-15], the word omnidirectional is a more technical synonym of the word *panorama*, which emerged in the late 18th century and is commonly used in the context of art. Puchberger invented the first panoramic camera in 1843 and inspired several innovators who constructed successors in the following years. These early omnidirectional cameras contained moving parts like

swing lenses or rotating lenses to capture a wide field of view. In addition, predecessors of wide-angle lenses were also introduced in the 19th century. A camera system that avoided moving parts and used a spherical lens filled with water was presented by Sutton in 1858.

In 1970, Rees was granted a patent for an omnidirectional camera system which used a hyperboloid mirror [Ree70]. The system captured the surface of the mirror and its resulting images could be transformed to normal perspective images, due to the uniqueness of the projection center. Figure 1.1 shows a modern setup of a hyperbolic mirror attached to a DSLR camera. Another catadioptric setup was introduced by Shree Nayar in 1988 [Nay88]. In contrast to Rees, Nayar captured the surface of two spheres in order to execute stereo vision algorithms.

Oh and Hall present a mobile robot [OH87] that uses a camera with a wide-angle fish-eye lens, followed by the work of Yagi and Kawato, who experiment with a mobile robot using a catadioptric camera [YK90, YKT91]. Yagi and Kawato call their system COPIS (standing for ***co****nic* ***p****rojection* ***i****mage* ***s****ensor*), which is used for obstacle avoidance. Another robot equipped with an omnidirectional image sensor is described by Thomas Geb in [Geb03]. Here, a reflective hemisphere is imaged for tele-operation.

Similar to the earlier work of Nayar in 1988, Gluckman, Nayar and Thoresz built an omnidirectional stereo-sensor by stacking two catadioptric cameras on top of each other [GNT98]. Their method produced a disparity map projected onto a cylindrical surface. More recent works on binocular omnidirectional vision systems are presented by Arican and Frossard [AF07] and Goto et al. [GYK+11]. Both systems align two cameras horizontally to construct a disparity map or reconstruct 3D points. Luo et al. [LHS+07] elaborate on a less common setup, in which a single camera captures two mirrors that are aligned behind one another.

Especially in the last two decades, more systems have been proposed and most of them are based on one of the configurations described by Nayar and Baker [Nay97, BN99] or Yagi [Yag99]. Ishiguro [Ish98] also addresses various catadioptric camera setups, while emphasizing low-cost solutions.

Experiments that were performed in the course of this thesis employed synthetic and real datasets. The V.360° by VSN Mobil has been used to capture own sequences, while the hardware used in external datasets is described in the relevant publications [SEE+12, GLU12, SSG14].

The last two sections briefly reviewed related concepts and hardware and placed this thesis into context. Subsequent chapters explain mathematical representations for omnidirectional cameras and discuss the development of a new monocular SLAM-system.

1.3 List of symbols

Geometric transforms

$\boldsymbol{R}$	Rotation matrix
$\boldsymbol{t}$	Translation vector or baseline
$\boldsymbol{T}$	Affine transformation

Coordinates

x	Real number:	$x \in \mathbb{R}$
$\boldsymbol{x}$	Real vector:	$x \in \mathbb{R}^n$
$\hat{\boldsymbol{x}}$	Normalized vector:	$\hat{\boldsymbol{x}} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$
$\boldsymbol{x}_c$	3D camera coordinate:	$(x, y, z, 0)^\top \in \mathbb{R}^4$ or $(x, y, z)^\top \in \mathbb{R}^3$
$\boldsymbol{u}$	Image coordinate:	$(du, dv, d)^\top \in \mathbb{P}^2$ or $(u, v)^\top \in \mathbb{R}^2$
C	Coordinate of camera frame:	$(x, y, z, 1)^\top \in \mathbb{R}^4$ or $(x, y, z)^\top \in \mathbb{R}^3$

Matrices

$\boldsymbol{K}$	Camera calibration matrix, pinhole camera model
$\boldsymbol{P}$	Camera matrix $= KT \in \mathbb{R}^{3\times4}$

$$[\mathbf{a}]_\times \stackrel{\text{def}}{=} \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

Parameters

$\lambda_i = c$	Configurable threshold with default value c
$\Lambda_i = c$	Configurable parameter with default value c

Chapter 2

Background

Visual SLAM systems consist of a variety of components, each of them providing different techniques and algorithms. Even though these components are customized for the special needs of chosen methods, some central tasks – such as the representation of motion – are indispensable, regardless of design choices. This chapter addresses fundamental, mainly mathematical, concepts that are relevant for most visual SLAM systems. All of the topics presented in this chapter will be revisited – directly or indirectly – afterward. As a profound discussion of every involved method is out of the scope of this thesis, more thorough literature is referred to where appropriate.

2.1 Representing motion

The position and orientation of a robot, referred to as *pose* of the robot, as well as its movement can be expressed using rigid-body motion. Rigid-body motion in $\mathbb{R}^n$ is commonly modeled by the set of matrices that corresponds to the special Euclidean group $\mathbf{SE}(n+1)$. Those matrices are of the following form:

$$\boldsymbol{M} \in \left\{ \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} \;\middle|\; \boldsymbol{R} \in \mathbf{SO}(n) \text{ and } \boldsymbol{t} \in \mathbb{R}^n \right\}$$

Where $\boldsymbol{R}$ is an element of the special orthogonal group $\mathbf{SO}(n)$, a common *Lie group*, which represents rotation, and $\boldsymbol{t} \in \mathbb{R}^n$ represents translation. Such transformation matrices can be applied to points and vectors in homogeneous coordinates. One major drawback of this representation is, however, that rotations are described by $n \times n$ matrices, although they have only n degrees of freedom. This makes it difficult to optimize them due to the high dimensionality and constraints,

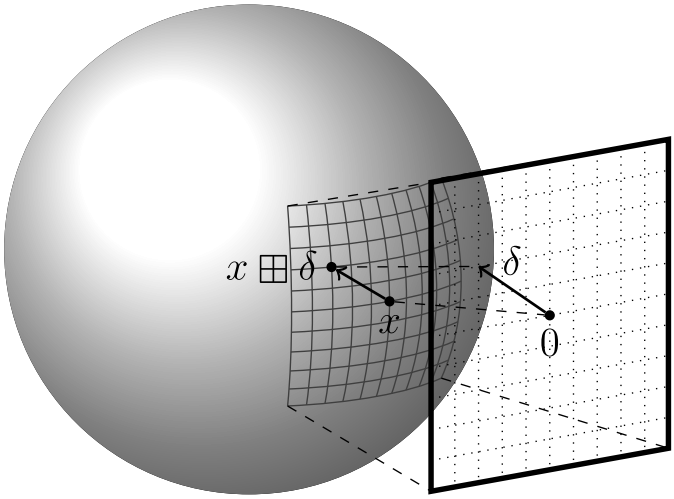


Figure 2.1: Illustration of the local mapping from a sphere surface to the tangent plane. The operator $\boxplus$ usually denotes an addition in tangent space, followed by a mapping back to the manifold: $\boxplus : \mathcal{M} \times \mathbb{R}^n \longmapsto \mathcal{M}$, where $x \in \mathcal{M}$ and $\delta \in \mathbb{R}^n$. Courtesy of Christoph Hertzberg, René Wagner, Udo Frese and Lutz Schröder, University of Bremen. Published in [HWFS11] as Figure 1. A detailed introduction of box operators is given in [Her08].

such as $\det(\boldsymbol{R}) = 1$. Hence, it is useful to have a more compact representation of the special Euclidean group. While it is possible to express rotations using quaternions, a more generic solution is offered by the *Lie algebra*, which is discussed in the following section.

2.2 Manifolds and Lie Groups

Manifolds and *Lie groups* are topological spaces that are often encountered in physics or robotics. While their structure is discussed rather briefly here, a more thorough introduction is given by Hauke Strasdat [Str12a] or textbooks such as [Gal11, Sti08]. Manifolds are spaces that locally behave Euclidean, but potentially behave differently in a global perspective. Figure 2.1 illustrates a sphere which is a common example of manifolds. Locally, the sphere behaves like a plane, the *tangent plane*, whereas the global behavior is non-linear.

Lie groups are differentiable manifolds so that group operations (multiplication and inversion) are smooth maps. This allows constructing the *tangent space* at each point of the manifold. The tangent space contains all vectors that tangentially pass through the point of interest. It turns out that the tangent space at the identity can be used to construct each element of the Lie group by repeating an infinitesimal transformation. This becomes clearer, when regarding the special orthogonal Lie

group $\mathbf{SO}(3)$ prototypically. Elements of this group represent rotations in 3D. An infinitesimal element of the tangent space at the identity expresses an infinitesimal change in rotation. When such an element is chosen correctly and the infinitesimal changes in rotation are accumulated often enough, then each rotation $\in \mathbf{SO}(3)$ can be constructed.

Example: Assume a rotation $\boldsymbol{R}(\alpha)_z$ of an arbitrary angle α around the z-axis is given, then the rotation matrix and the infinitesimal first order approximation are written as:

$$\boldsymbol{R}(\alpha)_z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$\boldsymbol{R}(\mathrm{d}\alpha)_z = \begin{bmatrix} 1 & -\mathrm{d}\alpha & 0 \\ \mathrm{d}\alpha & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I + \mathrm{d}\alpha \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = I + \mathrm{d}\alpha\hat{\boldsymbol{\omega}}_z \tag{2.2}$$

Now, it is possible to reconstruct the original rotation by the accumulation of infinitesimal rotational changes, which results in the *exponential mapping*:

$$\boldsymbol{R}(\alpha)_z = (I + \mathrm{d}\alpha\hat{\boldsymbol{\omega}}_z)^{\infty} = \lim_{n\to\infty} (I + \frac{\alpha\hat{\boldsymbol{\omega}}_z}{n})^n = \exp(\alpha\hat{\boldsymbol{\omega}}_z) \tag{2.3}$$

Equation 2.3 shows that it is possible to represent rotations around the z-axis through a scaled skew symmetric matrix $\alpha\hat{\boldsymbol{\omega}}_z$. This scheme can be generalized to rotations around arbitrary axes by adjusting the matrix accordingly:

$$\hat{\boldsymbol{\omega}} = [\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.4}$$

In the generalized case, $\hat{\boldsymbol{\omega}}$ is still a skew symmetric matrix and is called *generator of infinitesimal transformations*. Generators are elements of the tangent space, here of $\mathbf{SO}(3)$, and they form their own algebra, the *Lie algebra*, here $\hat{\boldsymbol{\omega}} \in \mathfrak{so}(3)$. The multiplication of the $\mathfrak{so}(3)$ algebra is known as *Lie bracket* and is given by:

$$[\,\cdot\,,\,\cdot\,] : \mathfrak{so}(3) \times \mathfrak{so}(3) \longmapsto \mathfrak{so}(3)\,, \quad [\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}] := \hat{\boldsymbol{w}}\hat{\boldsymbol{v}} - \hat{\boldsymbol{v}}\hat{\boldsymbol{w}} \tag{2.5}$$

So far, the Lie group $\mathbf{SO}(3)$ and the related Lie algebra $\mathfrak{so}(3)$ were presented. Equation 2.3 showed that there is an exponential mapping from elements of $\mathfrak{so}(3)$ to elements of $\mathbf{SO}(3)$. The inverse mapping is given straightaway by the logarithm. Applying the tangent space $\mathfrak{so}(3)$ instead of $\mathbf{SO}(3)$ should be considered due to the following reasons:

1. While $\mathfrak{so}(3)$ is linear, $\mathbf{SO}(3)$ is not.

2. In contrast[1] to elements of $\mathbf{SO}(3)$, elements of $\mathfrak{so}(3)$ are parametrized minimally and unconstrained. The unconstrained representation without singularities is especially useful for optimization purposes.

Like $\mathbf{SO}(3)$, the group of rigid transformations $\mathbf{SE}(3)$, as well as the group of similarity transformations $\mathbf{Sim}(3)$, are also Lie groups and are associated with Lie algebras in the same manner. While $\mathfrak{se}(3)$ contains elements of the tangent space of $\mathbf{SE}(3)$, $\mathfrak{sim}(3)$ consists of elements of the tangent space of $\mathbf{Sim}(3)$. In the context of pose graph optimization, $\mathfrak{sim}(3)$ is probably the most interesting transformation representation, since it embodies scaling. As explained later, scale drift is a major source of error in the presence of monocular sensors. Hence, explicitly addressing scale changes during optimization is advantageous.

2.3 Optimization

Virtually all computer vision methods encounter some sort of uncertainty, since the mere process of performing sensor measurements involves ambiguities. Non-synthetic images, for example, may exhibit a substantial degree of noise and quantization inaccuracies, resulting in contradicting measurements. Hence, algorithms try to find solutions that fit the data best, as a perfect fit is not possible in general. As soon as various solutions are feasible, the task of finding the best of them is an optimization problem. While there are algorithms that perform optimization rather implicitly, such as the Hough transform for line fitting, others make explicit use of optimization techniques. In order to design new methods, it is crucial to have an understanding of appearing uncertainties and to know which techniques are available for error reduction.

The general optimization problem can be expressed as follows. Given an *objective function* $f(\boldsymbol{x})$ of a parameter vector $\boldsymbol{x}$, the goal is to find the global minimizer $\boldsymbol{x}^*$:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in \mathbb{S}}{\operatorname{argmin}} f(\boldsymbol{x}) \tag{2.6}$$

Where $\mathbb{S}$ is the space of solutions and $\boldsymbol{x}$ might have to fulfill constraints for belonging to this space. Assume that $\boldsymbol{x}$ parametrizes a line in an image: In this prototypical example, the function f could assign each line configuration an error based on observations, i.e. the sum of square distances of observed points to that

[1] More precisely, the Euler angles $\mathbf{SO}(3)$ representation introduces singularities in the form of *gimbal locks*, and the $\mathbb{R}^{3\times3}$ matrix representation is redundant and entails constraints.

line. Then, $\boldsymbol{x}^*$ would be the line parametrization which fits the point observations best.

The following sections briefly review techniques to perform optimization, as in Equation 2.6, with focus on methods that have successfully been applied to computer vision. Throughout this chapter, it is assumed that objective functions are continuous and differentiable, leading to the methods presented. The notation used in the next chapter is based on the textbooks [Pri12, Tre13, NW06][2].

2.3.1 Maximum likelihood and Maximum a posteriori

As already mentioned, optimization techniques are employed due to different sources of uncertainty. Since uncertainty often occurs in a structured way, it is possible to find probabilistic formulations. *Bayes' rule* specifies a relation of joint probabilities that allows the development of those formulations. According to Bayes' rule, when two random variables $\boldsymbol{x}$ and $\boldsymbol{z}$ are given, the conditional probability or density $\Pr(\boldsymbol{x}|\,\boldsymbol{z})$ can be formulated as:

$$\Pr(\boldsymbol{x}|\,\boldsymbol{z}) = \frac{\Pr(\boldsymbol{z}|\,\boldsymbol{x})\Pr(\boldsymbol{x})}{\Pr(\boldsymbol{z})} \tag{2.7}$$

By convention, the term $\Pr(\boldsymbol{x}|\,\boldsymbol{z})$ is referred to as *posterior*, $\Pr(\boldsymbol{z}|\,\boldsymbol{x})$ as *likelihood*, $\Pr(\boldsymbol{x})$ as *prior* and $\Pr(\boldsymbol{z})$ as *evidence*. Imagine that $\boldsymbol{z}$ expresses sensor measurements, or point observations as in the earlier example, and $\boldsymbol{x}$ expresses a parametrization that is to be optimized; then $\Pr(\boldsymbol{x}|\,\boldsymbol{z})$ represents the post-observation probability or density of $\boldsymbol{x}$, $\Pr(\boldsymbol{z}|\,\boldsymbol{x})$ the likelihood of a measurement given $\boldsymbol{x}$ and $\Pr(\boldsymbol{x})$ represents prior knowledge of $\boldsymbol{x}$. The term $\Pr(\boldsymbol{z})$ is commonly dropped because it is independent of the parameter vector $\boldsymbol{x}$, and a scaling of $\Pr(\boldsymbol{z})^{-1}$ does not change the location of the maximizer of $\Pr(\boldsymbol{x}|\,\boldsymbol{z})$. While Equation 2.6 suggests minimizing the error related to a parameter vector, it is also possible to maximize the probability of a parameter vector, given measurements.

[2] In [Pri12] Simon Prince denotes *probability mass functions* as well as *probability density functions* with Pr, which is rather uncommon. The advantage of this notation is, however, that the discrete and continuous case do not have to be addressed separately, as in Equation 2.7, for example. After all, probability *mass* and *density* are strongly related. This notation has been adopted.

Assuming the independence of $n \in \mathbb{N}^+$ measurements $\boldsymbol{z}_1..\boldsymbol{z}_n$ and dropping the dominator yields the *maximum a posteriori* (MAP) fitting:

$$\begin{aligned} \boldsymbol{x}^* &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \left[\Pr(\boldsymbol{x} |\, \boldsymbol{z}_1..\boldsymbol{z}_n) \right] \\ &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \left[\frac{\Pr(\boldsymbol{z}_1..\boldsymbol{z}_n |\, \boldsymbol{x}) \Pr(\boldsymbol{x})}{\Pr(\boldsymbol{z}_1..\boldsymbol{z}_n)} \right] \\ &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} \Pr(\boldsymbol{z}_i |\, \boldsymbol{x}) \Pr(\boldsymbol{x}) \right] \end{aligned} \tag{2.8}$$

Should prior information be unavailable, a uniform distribution of $\Pr(\boldsymbol{x})$ can be assumed, leading to the *maximum likelihood* (ML) estimator:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} \Pr(\boldsymbol{z}_i |\, \boldsymbol{x}) \right] \tag{2.9}$$

As noted by Triggs el al. [TMHF00], ML estimators can also include prior information using additional observations. Therefore, the distinction between observation and prior is rather terminological. In order to compute the best ML parameter estimate $\boldsymbol{x}^*$, the derivative of Equation 2.9 has to be set to zero, which results in expensive computations. To ease the computation, the related *log-likelihood* method can be applied. The equation

$$\begin{aligned} \boldsymbol{x}^* &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \left[\ln \big(\prod_{i=1}^{n} \Pr(\boldsymbol{z}_i |\, \boldsymbol{x}) \big) \right] \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \left[- \ln \big(\prod_{i=1}^{n} \Pr(\boldsymbol{z}_i |\, \boldsymbol{x}) \big) \right] \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \left[- \sum_{i=1}^{n} \ln(\Pr(\boldsymbol{z}_i |\, \boldsymbol{x})) \right] \end{aligned} \tag{2.10}$$

yields the same result as Equation 2.9, since the extremes do not change after applying the monotonous logarithm function.

Application. Assume a model is parametrized by $\boldsymbol{x}$ and the input $\boldsymbol{z'}$ is mapped to an observation $\boldsymbol{z}$ under Gaussian noise $\varepsilon \sim \mathcal{N}(0, \Omega)$:

$$\boldsymbol{z} = g(\boldsymbol{x}, \boldsymbol{z'}) + \varepsilon \tag{2.11}$$

Where the model $g(\boldsymbol{x}, \boldsymbol{z'})$ could again represent a line in an image with observations $\boldsymbol{z}$ normally distributed around the line. Given the model of Equation 2.11, it is possible to express $\Pr(\boldsymbol{z}_i |\, \boldsymbol{x})$ as multivariate normal distribution with mean $\boldsymbol{z}_i$:

$$\Pr(\boldsymbol{z}_i |\, \boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^p \det(\Omega)}} \exp\left(-\frac{1}{2} (\boldsymbol{z}_i - g(\boldsymbol{x}, \boldsymbol{z}'_i))^T \Omega^{-1} (\boldsymbol{z}_i - g(\boldsymbol{x}, \boldsymbol{z}'_i)) \right) \tag{2.12}$$

Here, Ω denotes the covariance matrix of the normal distribution and p the dimension of $\boldsymbol{z}_i$. Combining Equation 2.10 and 2.12 leads to:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left[n \ln \sqrt{(2\pi)^p \det(\Omega)} + \frac{1}{2} \sum_{i=1}^{n} (\boldsymbol{z}_i - g(\boldsymbol{x}, \boldsymbol{z}'_i))^T \Omega^{-1} (\boldsymbol{z}_i - g(\boldsymbol{x}, \boldsymbol{z}'_i)) \right]$$

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{i=1}^{n} (\boldsymbol{z}_i - g(\boldsymbol{x}, \boldsymbol{z}'_i))^T \Omega^{-1} (\boldsymbol{z}_i - g(\boldsymbol{x}, \boldsymbol{z}'_i)) \right] \tag{2.13}$$

The structure of this problem is of a special form, which is called *least squares optimization* and will be covered in the next section. Before this, some remarks are owed:

1. It is not always appropriate to model noise by a normal distribution. Hence, not every maximum likelihood problem is a least squares problem.

2. Noise cannot always be added linearly. For instance, instead of adding noise linearly on Lie groups, one should consider using the Lie algebra.

3. The use of Bayes' rule is attractive, because representing $\Pr(\boldsymbol{x} | \boldsymbol{z})$ is problem-specific and the surface of this function is possibly (much) less uniform than the one of $\Pr(\boldsymbol{z} | \boldsymbol{x})$.

2.3.2 Least Squares

Optimization problems with the following structure:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} f(\boldsymbol{x}) = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} \|r_i(\boldsymbol{x})\|^2 \tag{2.14}$$

are referred to as *least-squares problems*, since their purpose is to reduce the sum of square values returned by the *error function* $r_i(\boldsymbol{x})$. Please note that squaring the norm is equal to summing squared entries of $r_i(\boldsymbol{x})$ component-wise. It is common that the error function expresses the difference between a measurement and an expectation. This difference is also called *residual* and in accordance with the last section could be written as:

$$r_i(\boldsymbol{x}) = \boldsymbol{z}_i - g(\boldsymbol{x}, \boldsymbol{z}'_i) \tag{2.15}$$

Assuming that the covariance matrix Ω is the identity matrix, Equation 2.13 already matches a least squares problem. In practice, the least squares problem is often described by:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{i=1}^{n} r_i(\boldsymbol{x})^\top \Omega^{-1} r_i(\boldsymbol{x}) \tag{2.16}$$

Where Ω^{-1} is called the information matrix. Expression 2.16, seemingly[3] more general than 2.14, is called the *generalized least squares*. If only diagonal entries of Ω^{-1} are unequal to zero, the problem is referred to as *weighted least squares*, as this corresponds to Equation 2.14 with arbitrary weighted components. Here, the scaling by $\frac{1}{2}$ has been dropped, as it does not effect the minimizer. Given the generalized least squares formulation, the minimization of the negative log-likelihood under Gaussian noise perfectly matches a least squares problem.

The advantage of least squares optimization is that the special structure can be exploited in order to derive faster or more stable algorithms. In the linear case, for example, where each $r_i(\boldsymbol{x})$ is linear, it is possible to compute $\boldsymbol{x}^*$ algebraically. The *Gauss-Newton* method is not restricted to the linear case and is presented in the next chapter.

2.3.3 Gauss-Newton

The idea behind the Gauss-Newton method is simple: Given an initial guess $\boldsymbol{x}^0$ near $\boldsymbol{x}^*$, assume that the error functions are linear and compute the best fit $\boldsymbol{x}^1$ algebraically using the linear model. Since the assumption of having linear error functions does not hold in general, $\boldsymbol{x}^1 = \boldsymbol{x}^*$ will usually not satisfy. By repeating the linearization step arbitrarily often, further successors of $\boldsymbol{x}^0$ are computed until the solution is good enough. The first order Taylor approximation of $r_i(\boldsymbol{x})$ is:

$$r_i(\boldsymbol{x} + \boldsymbol{\Delta x}) \approx r_i(\boldsymbol{x}) + \boldsymbol{J}_i \boldsymbol{\Delta x} \tag{2.17}$$

[3] In fact, generalized least square problems can be rearranged to ordinary ones by transforming the data. Being a covariance matrix, Ω is positive semi-definite, allowing the decomposition: $\sum r^\top \Omega^{-1} r = \sum r^\top (LL^\top)^{-1} r = \sum (L^{-1}r)^\top (L^{-1}r) = \sum \|(L^{-1}r)\|^2$

Where $\boldsymbol{J}_i$ is the Jacobian of the function r_i. Then the squared norm of the residual is written as follows[4]:

$$\begin{aligned} \|r_i(\boldsymbol{x}+\boldsymbol{\Delta x})\|^2 &= r_i(\boldsymbol{x}+\boldsymbol{\Delta x})^\top r_i(\boldsymbol{x}+\boldsymbol{\Delta x}) \\ &\approx (r_i(\boldsymbol{x})+\boldsymbol{J}_i\boldsymbol{\Delta x})^\top (r_i(\boldsymbol{x})+\boldsymbol{J}_i\boldsymbol{\Delta x}) \\ &= r_i^\top(\boldsymbol{x})r_i(\boldsymbol{x}) + r_i^\top(\boldsymbol{x})\boldsymbol{J}_i\boldsymbol{\Delta x} + (\boldsymbol{J}_i\boldsymbol{\Delta x})^\top r_i(\boldsymbol{x}) + (\boldsymbol{J}_i\boldsymbol{\Delta x})^\top \boldsymbol{J}_i\boldsymbol{\Delta x} \\ &= r_i^\top(\boldsymbol{x})r_i(\boldsymbol{x}) + 2r_i^\top(\boldsymbol{x})\boldsymbol{J}_i\boldsymbol{\Delta x} + \boldsymbol{\Delta x}^\top \boldsymbol{J}_i^\top \boldsymbol{J}_i\boldsymbol{\Delta x} \end{aligned} \tag{2.18}$$

Remarkably, the term $\boldsymbol{J}_i^\top \boldsymbol{J}_i$ in the last line of this equation is an approximation to the Hessian matrix. This means that solving Equation 2.18 for $\boldsymbol{\Delta x}$ performs a second order approximation using only first order information. To verify this, the separately computed Hessian writes:

$$H_i = 2\boldsymbol{J}_i^\top \boldsymbol{J}_i + 2\frac{\partial^2 r_i(\boldsymbol{x})}{\partial \boldsymbol{x}_j \partial \boldsymbol{x}_k} \tag{2.19}$$

Here j and k are indices for partial derivatives. Formulating a second order Taylor expansion of the squared norm term would result in an equation similar to 2.18, which exhibits the additional partial derivatives of Equation 2.19. Avoiding these second order derivatives is preferable, however, as their computation is demanding. Furthermore, they are less dominant than the $\boldsymbol{J}_i^\top \boldsymbol{J}_i$ part, especially for small residuals.

So far, the summation of residuals has been neglected. Combining Equation 2.14 and 2.18 yields

$$\begin{aligned} \boldsymbol{\Delta x}^* &\approx \underset{\boldsymbol{\Delta x}}{\operatorname{argmin}} \sum_{i=1}^{n} r_i^\top(\boldsymbol{x})r_i(\boldsymbol{x}) + 2r_i^\top(\boldsymbol{x})\boldsymbol{J}_i\boldsymbol{\Delta x} + \boldsymbol{\Delta x}^\top \boldsymbol{J}_i^\top \boldsymbol{J}_i\boldsymbol{\Delta x} \\ &= \underset{\boldsymbol{\Delta x}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[r_i^\top(\boldsymbol{x})r_i(\boldsymbol{x})\right] + 2\sum_{i=1}^{n} \left[r_i^\top(\boldsymbol{x})\boldsymbol{J}_i\right]\boldsymbol{\Delta x} + \boldsymbol{\Delta x}^\top \sum_{i=1}^{n} \left[\boldsymbol{J}_i^\top \boldsymbol{J}_i\right]\boldsymbol{\Delta x}\,, \end{aligned} \tag{2.20}$$

which is quadratic in $\boldsymbol{\Delta x}$. After setting the derivative to zero, $\boldsymbol{\Delta x}^*$ is computed by solving:

$$\sum_{i=1}^{n} \left[\boldsymbol{J}_i^\top \boldsymbol{J}_i\right]\boldsymbol{\Delta x}^* = -\sum_{i=1}^{n} \left[r_i^\top(\boldsymbol{x})\boldsymbol{J}_i\right] \tag{2.21}$$

[4]Here the Gauss-Newton method for plain least squares is derived instead of the generalized least squares. Since the proceeding is exactly the same in both cases, the less distracting variant is presented.

In literature, the explicit notation of the summation is commonly avoided, which leads to the clearer form:

$$\boldsymbol{J}^\top \boldsymbol{J} \Delta \boldsymbol{x}^* = -r^\top \boldsymbol{J} = -\boldsymbol{J}^\top r \tag{2.22}$$

As mentioned by Grisetti et al. [GKSK11], the summed matrix $\boldsymbol{J}^\top \boldsymbol{J}$ is usually sparse by construction. Hence, it can be efficiently solved using methods like Cholesky factorization.

Problems of the Gauss-Newton method are that it is only stable for good initial estimates $\boldsymbol{x}^0$ and that neither convergence nor an improvement with each iteration is guaranteed. Two improvements in order to overcome these drawbacks lead to the Levenberg-Marquardt algorithm.

2.3.4 Levenberg-Marquardt

The *gradient descent* method offers an intuitive approach for optimization. Having any continuous differentiable function, the negative gradient will point towards a local or even global minimum of this function. Iteratively taking small enough steps in this direction will therefore lead to a minimizer. Computing the gradient of the residuals yields

$$\bigtriangledown \frac{1}{2} \sum_{i=1}^{n} \|r_i(\boldsymbol{x})\|^2 = -\boldsymbol{J}^\top r \quad , \tag{2.23}$$

using the same notation of Equation 2.22. While being rather stable, the convergence rate of gradient descent is known to be small. Both Levenberg [Lev44] and Marquardt [Mar63] suggested combining the Gauss-Newton method with gradient descent for least squares problems by performing gradient descent steps at the beginning or when convergence is low and Gauss-Newton otherwise. This can be accomplished using the following equation:

$$(\boldsymbol{J}^\top \boldsymbol{J} + \lambda \boldsymbol{I}) \Delta \boldsymbol{x}^* = -\boldsymbol{J}^\top r \quad , \tag{2.24}$$

where $\boldsymbol{I}$ is the unit matrix and λ a damping factor. This expression is equivalent to 2.22, when $\lambda = 0$. Yet, $\Delta \boldsymbol{x}^*$ develops into the same direction as the gradient for large λ, since the weight of $\boldsymbol{J}^\top \boldsymbol{J}$ decreases with increasing λ. This results in an interpolation between gradient descent and the Gauss-Newton method based on the value of λ. As described in [DS96], replacing the identity matrix with a positive diagonal matrix $\boldsymbol{D}^2$ is equivalent to scaling the components of the gradient. To

improve the convergence speed at situations where the gradient is small but λ and the curvature are high, Marquardt introduced a change of Equation 2.24 to[5]:

$$(\boldsymbol{J}^\top \boldsymbol{J} + \lambda \text{diag}[\boldsymbol{J}^\top \boldsymbol{J}])\boldsymbol{\Delta x}^* = -\boldsymbol{J}^\top r \quad , \tag{2.25}$$

which is the essential equation of Levenberg-Marquardt optimization. Implementation details and choices for the damping parameter λ are discussed in the book [SW03] by Seber and Wild.

2.3.5 Robust error functions

Up to now, the optimization discussion emphasized a Gaussian noise model. There are scenarios, however, where *outliers* occur in addition to noise. Outliers are measurements associated with a high error, usually due to misinterpreted data. Mismatching image features, for instance, is a common source for outliers in the field of computer vision. The error function of Equation 2.14 is quadratic in the length of the residual. Therefore, few outlier measurements with high errors have a massive impact of the overall error, misleading the optimization process. This effect can be reduced by employing alternative error functions. Given the norm of a residual $\delta_i = \|r_i(\boldsymbol{x})\|$, the error was previously calculated by $C_s(\delta) = \delta^2$. Several alternatives to the squared error cost function are available, such as the *Huber function*:

$$C_h(\delta) := \begin{cases} \delta^2 & if\ \delta < \lambda_b \\ 2b\delta & else \end{cases} \tag{2.26}$$

Here, the constant λ_b has to be chosen according to the outlier threshold. Like a squared error function, the Huber function is quadratic in δ for small errors, but linear once a threshold is passed, which effectively reduces the impact of outliers. Being convex, the Huber function does not introduce additional minima to the objective function. Figure 2.2 plots both the Huber and quadratic cost function.

In [HZ04, p.616-622], Hartley and Zissermann compare several error functions and explain how to apply them in least square optimization. While the Levenberg-Marquardt method still expects problems with a quadratic cost function, inserting weighted residuals allows using arbitrary cost functions:

$$\begin{aligned} w_i^2 \|r_i(\boldsymbol{x})\|^2 &= C(\|r_i(\boldsymbol{x})\|) \\ w_i &= \frac{\sqrt{C(\|r_i(\boldsymbol{x})\|)}}{\|r_i(\boldsymbol{x})\|} \end{aligned} \tag{2.27}$$

[5] Marquardt proposed this equation in a different but equivalent form. Since the style adopted here was introduced by [FAH71], some authors claim that he suggested to use $\text{diag}[\boldsymbol{J}^\top \boldsymbol{J}]$ instead of $\boldsymbol{I}$.

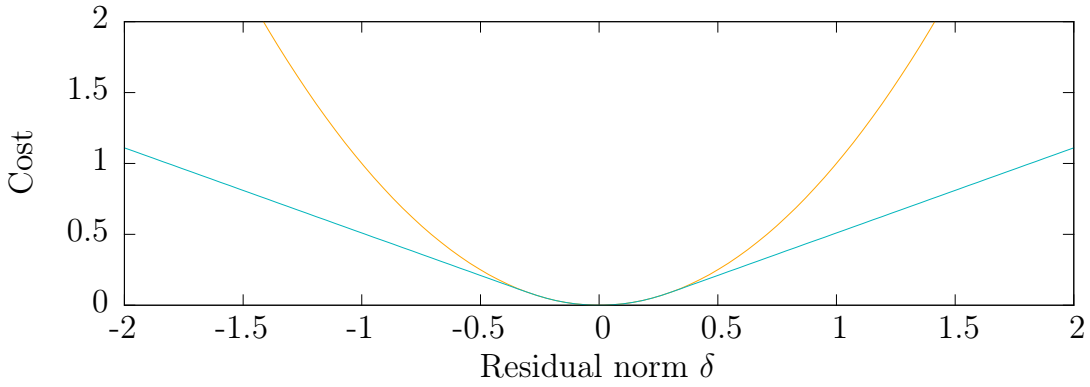


Figure 2.2: Comparison of the quadratic (orange) and Huber (turquoise) cost function with a threshold $\lambda_b = 0.3$. In contrast to the quadratic cost-function, the Huber function behaves linearly for $\delta \geq 0.3$.

In this equation, w_i is a weight for the residual $r_i(\boldsymbol{x})$ and $w_i^2 \|r_i(\boldsymbol{x})\|^2$ is the new cost function which suffices a least squares structure. In order to perform a Gauss-Newton or Levenberg-Marquardt iteration, first, the original residuals $r_i(\boldsymbol{x})$ are computed as usual, followed by computing the incremental step using the weighted cost function of Equation 2.27.

2.3.6 Graph-based optimization

Various optimization problems, especially those that are SLAM related, exhibit a (hyper-)*graph* structure that can be used for optimization. An influential work on that matter was presented by Lu and Milios in 1997 [LM97], where the graph is referred to as *network of pose relations*. The idea was to represent each pose of a robot as a node in a network connected by links that represent relative motion, and to perform global optimization on this network. Although the concept of constructing a graph of poses dates back to 1986 [SC86], it was previously common to employ sequential mapping only, for example by matching laser-range scans of two consecutive robot locations. Neglecting global information, however, results in accumulated errors that can severely interfere with the mapping process. This is illustrated in Figure 2.3, where the course created using relative motion only, is compared to a globally optimized trajectory. As already mentioned above, a graph exhibits nodes and edges, corresponding to robot poses and relative motion respectively. In more general terms, nodes of the graph form vectors of parameters to be estimated or optimized, and edges form constraints between the nodes. Given a cost function that assigns an error to an edge based on how good the constraint is fulfilled, it is possible to employ the Levenberg-Marquardt minimizer to execute the

(a) Path from the odometry readings. **(b)** Recovered path, correct within $\pm 1m$.

Figure 2.3: Using only odometry readings for localization results in accumulated errors, as the path in the left image shows. The magnitude of the error becomes obvious when comparing the left to the right image, which shows the true trajectory. Both images display the course from the Victoria Park dataset. Courtesy of José Guivant and Eduardo Nebot, Australian Centre for Field Robotics and Michael Montemerlo, Stanford University. Published in [TBF05] as Figure 12.14 and 12.15.

graph optimization. When optimizing 3D robot poses, each pose has 6 or 7 degrees of freedom depending on whether scale is considered in addition to rotation and translation. As a consequence, the number of stacked parameters to be optimized increases linearly with the number of poses. For instance, having 30 poses with 7 degrees of freedom leads to a non-linear optimization with 210 parameters. The number is considerably higher, when observed data is optimized as well. Figure 5.2 illustrates a graph structure, which arises when bundle-adjustment is performed using graph optimization. In order to compute the high amount of parameters that is usually involved in graph optimization efficiently, the sparsity of the problem has to be exploited. Because a single constraint only affects the parameters connected to the edge, it contributes to few entries of the Jacobian and Hessian matrix. Torr et al. [TMHF00] describe how the Schur complement allows to exploit this property. The general graph-based optimization back-end (g^2o) used in the implementation was presented by Kümmerle et al. in 2011 [GKSK11]. It supports various types of graphs and covers domain-specific implementations for the user.

2.3.7 RANSAC

RANSAC, introduced by Fischler and Bolles in 1981 [FB81], stands for ***Ran****dom* ***Sa****mple* ***C****onsensus* and describes a procedure to perform model to data fitting. In this regard, it can be compared to the least squares method described earlier, since both approaches allow to estimate the parameters of a model with the help of data. RANSAC, however, specifically aims at being robust against outlier in the data by performing the following steps:

1. Select a random minimal subset of the data, which allows to compute model parameters.
2. Fit a model to the data-subset.
3. For each datum, compute the divergence to the model and count how many times the divergence is lower than a threshold (inlier count).

These steps are repeated many times, and the resulting model is either the one with the highest inlier count, or computed – with least squares, for instance – using all the inliers of the model with the highest inlier count. When the amount of outliers can be estimated, it is possible to calculate the number of iterations required to achieve a predefined certainty with which an outlier-free sample was chosen. Let ϵ be the probability that a datum is an outlier and s be the size of the minimal subset, then $(1-\epsilon)^s$ results in the probability that a subset is free of outliers; and $(1-(1-\epsilon)^s)^N$ is the probability that after sampling N times, each sample contains an outlier. Hence, N has to be chosen in a way that $(1-(1-\epsilon)^s)^N = 1-p$ is fulfilled, where p is the desired predefined confidence. This leads to:

$$N = \frac{log(1-p)}{log(1-(1-\epsilon)^s)} \tag{2.28}$$

Here, N should be rounded up to get a confidence of at least p. The threshold mentioned in step 3 of the RANSAC-procedure can be estimated either empirically or by exploiting the error model of inlier observations. Given data with an additive Gaussian error model, the cumulative χ^2 distribution reflects the likelihood of the parameters used to fit the data. Concisely illustrated, this happens as follows: If a random variable $Z \sim \mathcal{N}(0,1)$ – an inlier observation – then $\chi^2 \sim Z^2$, which means that high deviations from 0 are unlikely. After measuring deviations between expected and observed data, the monotonically increasing cumulative χ^2 maps the probability of Z not being normally distributed to the deviations. Hence, the cumulative χ^2 maps a high probability to outliers, because their measurements are not normally distributed. Hartley and Zisserman suggest [HZ04, p.119] to compute the RANSAC-threshold by rejecting 5% of the normally distributed inliers.

This way, the threshold coincides with the value that is mapped to 95% by the cumulative χ^2 distribution, and it can be computed if the variance of the Gaussian error model is known. Although 5% of the inliers are rejected using this approach, Equation 2.28 remains valid, since the true amount of outliers is unaltered.

A more detailed introduction to RANSAC is available in [HZ04, p.116-124]. There are many derivatives of the original RANSAC procedure and a comprehensive overview of them is given by Matas and Chum [MC11]. The method used in the implementation is based on MSAC (***M**-estimator **Sa**mple and **C**onsensus*) [TZ00]. It differs from the original RANSAC algorithm in the scoring of models. While the original method scores a sample based on the inlier count, MSAC takes the quality of inliers into account. This is done by examining the error e of each datum, given an estimated model. The model with the minimum overall-error C, which is computed as follows, is chosen:

$$C = \sum_i p(e) \tag{2.29}$$

$$p(e) = \begin{cases} e^2 & if \;\; e^2 < \lambda_t^2 \\ \lambda_t^2 & else \end{cases}$$

Where λ_t is the RANSAC threshold.

Chapter 3

Camera models

The main subject of computer vision is to design computational methods to infer knowledge from visual sensors. In order to do so, it is necessary to derive mathematical representations for the involved sensors. There are numerous camera models that express these representations. An exhaustive comparison of camera models and calibration methods is provided by Sturm et al. [Stu10]. In general, camera models try to formulate *forward* and *back-projection* algebraically, where forward-projection describes a mapping from Cartesian 3D coordinates to pixel coordinates, and back-projection a mapping from pixel coordinates to a Cartesian 3D direction[1]. Usually, both mappings differ in complexity. Most visual SLAM systems employ cameras that can be represented with the pinhole camera model. For this reason, the pinhole camera model is briefly reviewed in the next section, followed by omnidirectional camera models that are able to represent the hardware introduced in Section 8.4 and the datasets discussed in Chapter 9.

3.1 Pinhole Model

The central-projection or pinhole camera model assumes that optical rays are projected onto the image plane disregarding dioptric (lenses) or catoptric (mirrors) elements. World points are mapped on the image plane through the camera center C, also called the pinhole, as illustrated in Figure 3.1. Note that one could also flip the image plane in front of the camera, creating a virtual image plane with

[1]Please note that in the context of bundle-adjustment the term back-projection is used reversely. The direction of projection should be clear when read in context though.

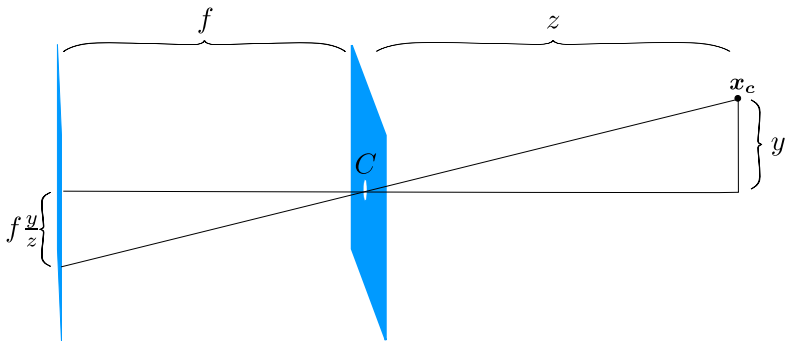


Figure 3.1: Side-view of the pinhole camera model. C is the camera centre, f the focal length, $\boldsymbol{x}_c$ a point in space and $f\frac{y}{z}$ the vertical image of $\boldsymbol{x}_c$.

the same properties. Given a 3D point in homogeneous coordinates, this mapping can be expressed as:

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{K} \boldsymbol{x}_c \\ \boldsymbol{u} &= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_c \end{aligned} \tag{3.1}$$

Where $\boldsymbol{u} = (wu, wv, w)$ is the projected point in homogeneous form and the Euclidean plane at $w = 1$ constitutes the image. In practice, more parameters related to the camera chip are involved in K. Additionally, it is necessary to transform world coordinates to camera coordinates. Extending Equation 4.2 accordingly yields:

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{KT} \boldsymbol{x} \\ \boldsymbol{u} &= \begin{bmatrix} \alpha_1 & s & u_0 & 0 \\ 0 & \alpha_2 & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{T} \boldsymbol{x} \\ \boldsymbol{u} &= \boldsymbol{P} \boldsymbol{x} \end{aligned} \tag{3.2}$$

Now, $\boldsymbol{x}$ is a point in homogeneous world coordinates, T the world to camera transformation matrix and P the compiled projection matrix. While α_1 and α_2 are related to the focal length and perform a scaling due to signal sampling, s is a called skew and compensates for potentially misaligned rows. u_0 and v_0 are translational components that are required to place the coordinate systems origin correctly. Both transformations K and P are invertible, but only up to scale. An in-depth discourse on the central-projection model can be found in [HZ04] or [Cor11]. Important properties of the pinhole camera model are:

1. Straight lines are preserved

2. Parallel lines intersect at a vanishing point
3. Conics are mapped to conics
4. Angles are not preserved

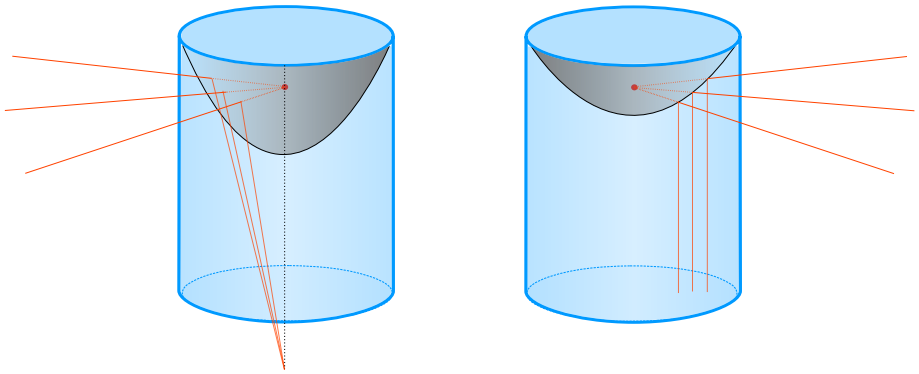
3.2 Omnidirectional Models

One major drawback of conventional camera systems is that they offer a limited field of view, whereas, in many applications, a large field of view is beneficial. Examples include motion tracking for robots or surveillance systems. In 1998, Baker and Nayar [BN98] showed that catadioptric vision systems – systems where traditional cameras are combined with mirrors – are well-suited for large field of view applications. They analyzed various catadioptric camera settings with different kinds of mirrors and concluded that neither planar nor conical nor spherical mirrors are able to enhance the field of view as long as an effective *single-view* camera is of interest. Instead, it is possible to use parabolic, hyperboloid or elliptical mirrors.[2] Single-view or *central cameras* are vision systems with a single center of projection. In [Nay97], Nayar lists advantages of central cameras. For instance, central cameras allow the generation of geometrically correct perspective images by back-projecting pixel values onto a plane. Further, in [SP02], Svoboda and Pajdla extend classical epipolar geometry to central catadioptric cameras. Using a central camera for computer vision is not a necessary requirement however. Most of the textbook [HKS08] is dedicated with non-central panoramic imaging, also covering topics like 3D reconstruction.

Figure 3.2 illustrates how rays of light behave in a parabolic and hyperboloid mirror setup. In both cases, the mirror exhibits a focus and rays of light directed to this point are either reflected orthographically, as with parabolic mirrors seen in Figure 3.2b, or directed towards another central point, as in Figure 3.2a. As a result, parabolic mirrors have to be used in combination with a telecentric lens in order to preserve a single effective view point. Since the application of telecentric lenses has been avoided studiously in experiments, hyperboloid mirrors in conjunction with regular camera systems constitute the standard catadioptric vision device.

In [BN98], Baker and Nayar analyze resolution properties of catadioptric cameras and conclude that the resolution is highest at the periphery of an omnidirectional image, as in Figure 1.1b. At the blind spot in the center of the figure, the camera is filming itself through the mirror, which is characteristic for central catadioptric vision system.

[2] This was independently derived by Svoboda et al. as described in [Svo00a]



(a) Camera with hyperboloidal mirror **(b)** Camera with parabolic mirror

Figure 3.2: Comparison of two popular catadioptric mirror types. While rays of light directed towards the focus of a hyperbolic mirror intersect after reflection, rays are oriented parallel to each other after being reflected by a parabolic mirror.

3.2.1 Hyperboloid Catadioptric Model

Analogously to the formulation of the pinhole camera model, it is possible to derive a mapping from world coordinates to image coordinates given a catadioptric camera system. This derivation has been performed by Tomáš Svoboda [Svo00a] and is outlined in the following.

Assume that the Cartesian camera coordinate system originates at the focus point of the hyperboloid – the z-axis aligned to the axis of the mirror. Given a point $\boldsymbol{x}_c = (x, y, z)^\top$ in camera coordinates, the mapping to pixel coordinates can be realized as follows:

$$\boldsymbol{u} = \boldsymbol{P} \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \\ 1 \end{pmatrix} \tag{3.3}$$

This actually performs two projections. The first projection maps $\boldsymbol{x}_c$ to the hyperboloid surface by multiplying the camera coordinate by a factor λ. The resulting vector is then transformed to a homogeneous coordinate and multiplied by a pinhole camera matrix $\boldsymbol{P}$ as in 3.1. λ can be computed by substituting a ray expression into a two-sheeted hyperboloid equation, with the hyperboloid expression being defined by:

$$\frac{(z+e)^2}{a^2} - \frac{x^2}{b^2} - \frac{y^2}{b^2} = 1 \tag{3.4}$$

Where $e = \sqrt{a^2 + b^2}$ is the mirror's eccentricity, and both a and b are mirror parameters. It has to be considered that the projection center of the pinhole camera has to coincide with the second focus of the mirror, which is located at $\boldsymbol{t} = (0, 0, 2e)^\top$. Otherwise, the reflection of a ray that emerges from the pinhole camera is not directed toward the focus of the mirror. A ray $\lambda \boldsymbol{x}_c$ that goes through the origin of the camera frame and $\boldsymbol{x}_c$, intersects the mirror at:

$$\lambda = \frac{b^2(-ez \pm a\|\boldsymbol{x}_c\|)}{b^2 z^2 - a^2 x^2 - a^2 y^2} \tag{3.5}$$

Using this equation in conjunction with Equation 3.3 yields the complete mapping from world points (in camera coordinates) to image coordinates. Equation 3.5 has multiple solutions though. Svoboda showed that one has to pick either the greater value of λ in case the results have different signs, or the smaller value in case both results have a positive sign. The third theoretical combination, namely that both results are negative, is not legitimate. The same approach from above also allows to calculate the back-projection of an image coordinate, given as:

$$\boldsymbol{P}^{-1}\boldsymbol{u} = \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 2e \end{pmatrix} \tag{3.6}$$

Where it is assumed that $\boldsymbol{P}^{-1}$ yields a homogeneous coordinate, which is transformed to the mirror's reference frame. Then, the intersection of this ray with the mirror is given by choosing λ correctly. The point of intersection concurrently corresponds to the direction of back-projection $\hat{\boldsymbol{x}}_c$, with respect to the coordinate frame at the focus. Again, λ can be computed by substituting the according ray expression in Equation 3.4.

An alternative way to formulate the mapping between image and world coordinates was given by Mičušík and Pajdla [MP03]. Exploiting the rotational symmetry, they introduced the non-linear function g that maps from pre-calibrated image coordinates to 3D-space:

$$\boldsymbol{x}_c \sim g(u, v) = (u, v, f(u, v))^\top \tag{3.7}$$

In this context, pre-calibrated means that the image coordinate is perfectly aligned with the mirror's z-axis and that horizontal mirror cross-sections are mapped to circles in the image. This is illustrated in Figure 3.3. Consequently, the image coordinate coincides with the first two components of the corresponding 3D-ray. The last component of the ray is given by the function f, which depends on the image coordinate. Thus, f incorporates the mirror design and adjusts the z-component of the back-projected ray. This way of modelling catadioptric camera systems will be revisited in the next sections. A detailed description of this formalism can be found in the PhD theses of Branislav Mičušík [Mic04] and Davide Scaramuzza [Sca08].

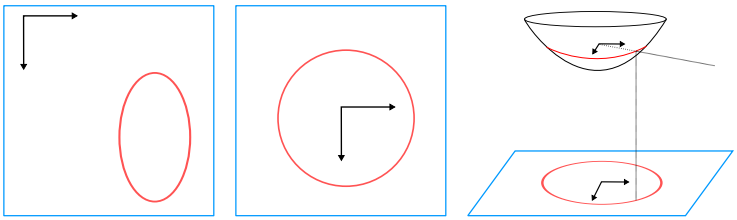


Figure 3.3: While (a) characterizes an image in pixel coordinates, the same pre-calibrated image is illustrated in (b). Both are related by an affine transformation due to digitizing and misalignments. (c) depicts the connection between the image space and the reference frame at the mirror's focus.

3.2.2 Unified Projection Model: Geyer and Daniilidis

Textbooks, such as [SNS11], refer to the model of Geyer and Daniilidis [GD00] as a landmark in omnidirectional computer vision. Geyer and Daniilidis introduced a unifying theory for central omnidirectional systems that allows the use of the same principles for hyperbolic-, elliptical- and parabolic-catadioptric, as well as traditional perspective, cameras. The authors proved that each of the aforementioned camera configurations can be modelled by projecting world points onto a unit sphere and successively onto a plane. This projection procedure is parametrized by the parameter[3] l. Adjusting this parameter accordingly results in a projection model that is isomorphic to one of the above. Figure 3.4 illustrates the nature of these projections.

Assume that the reference frame is located at the center of a unit circle that coincides with the mirror's focus (or one of the foci). The z-axis is again aligned with the major axis of the mirror. World points given in camera coordinates can be projected onto the sphere by:

$$\dot{\boldsymbol{x}} = \frac{\boldsymbol{x}_c}{\|\boldsymbol{x}_c\|} = (x, y, z)^\top \tag{3.8}$$

Given a second center of projection at $\boldsymbol{l} = (0, 0, -l)^\top$ on the z-axis, points $\dot{\boldsymbol{x}}$ are then projected onto a plane with distance 1 to the new projection center. This results in the coordinates:

$$\boldsymbol{u} = (\frac{x}{z+l}, \frac{y}{z+l}, 1)^\top = g(\dot{\boldsymbol{x}})^{-1} \quad , \tag{3.9}$$

[3]The original paper introduced two parameters, l and m. Here, for simplicity the notation of [SNS11] is used, which assumes that $l + m = 1$. The overall scheme remains the same.

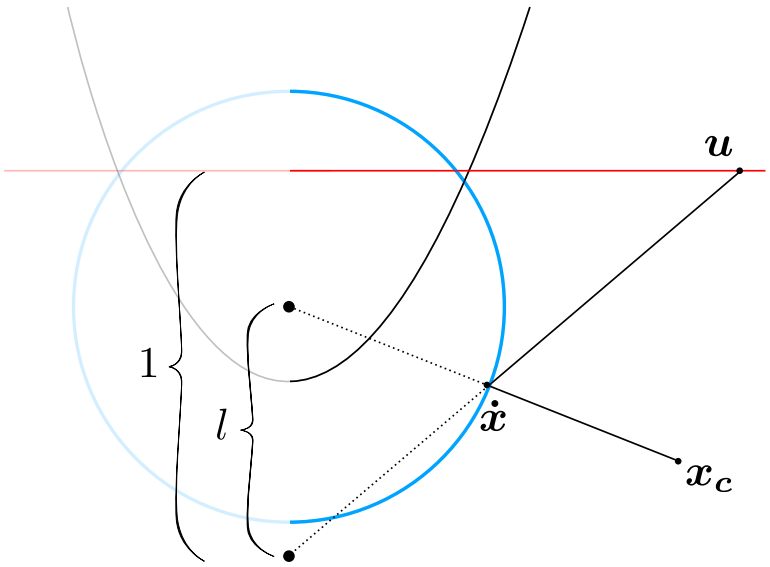


Figure 3.4: Visualization of the projection scheme presented by Geyer and Daniilidis. A point is firstly projected onto a sphere, which is highlighted in blue in this figure, and then projected onto a plane, which is highlighted in red.

which are relative to the second center of projection. A visualization of the projection scheme is given in Figure 3.4.

Performing equivalent projections, first onto a hyperboloid before projecting onto the plane, also yields Equation 3.9 when $l = 2e(4e^2 + 4p^2)^{-\frac{1}{2}}$, where e is the eccentricity and p a quarter of the latus rectum of the mirror. The latus rectum equals the diameter of the mirror at $z = 0$. As described in [SNS11], the mapping from sphere to plane is bijective, hence, the back-projection is given by:

$$g(\boldsymbol{u}) = g(x, y) = \begin{pmatrix} x \\ y \\ 1 - l\frac{x^2+y^2+1}{l+\sqrt{1+(1-l^2)(x^2+y^2)}} \end{pmatrix} \tag{3.10}$$

Note that this expression conforms to the formulation of Mičušík and Pajdla [MP03]. The third component of g is equivalent to the function f in Equation 3.7.

3.2.3 Unified Projection Model: Scaramuzza

The possibly most widely used calibration method for omnidirectional cameras was introduced by Scaramuzza et al. in 2006 [SMS06a], and an improved version

was published by the same authors shortly afterward [SMS06b]. The projection model of Scaramuzza et al. works in the same manner as the one of Geyer and Daniilidis, but the function g is approximated using Taylor series expansion:

$$g(x,y) = \begin{pmatrix} x \\ y \\ a_0 + a_2 p^2 + ... + a_n p^n \end{pmatrix} \quad , \tag{3.11}$$

where $p = \sqrt{x^2 + y^2}$. In [SMS06b], Scaramuzza et al. realized that it is possible to drop the coefficient a_1. This is due to the observation that a tangent at the mirror[4] surface at $p = 0$ is perpendicular to the z-axis. Hence, the derivative of the polynomial at $p = 0$ must be 0, which forces the constraint $a_1 = 0$.

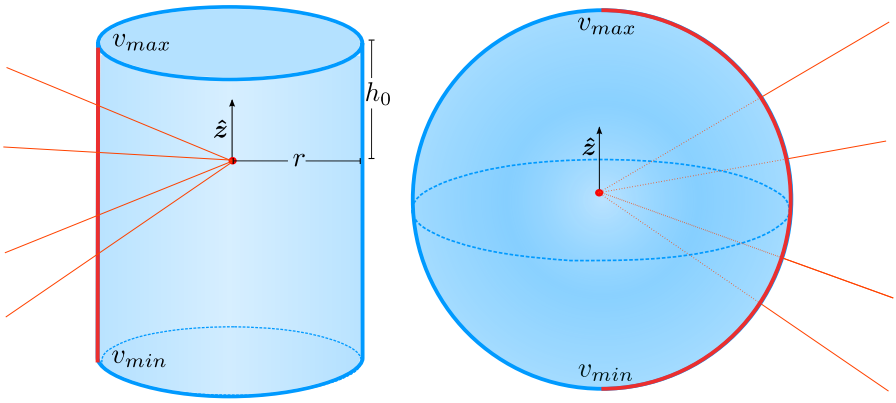
3.2.4 Projecting onto Spheres and Cylinders

Especially for visualization purposes, it is common to back-project image intensity values onto a spherical or cylindrical surface. Given a normalized vector $\boldsymbol{x}_c \sim g(x,y)$ in Euclidean 3D coordinates, the projection can be applied by expressing $\boldsymbol{x}_c$ in spherical or cylindrical coordinates respectively and by scaling the radius and, if applicable, the elevation accordingly:

$$\begin{aligned}
\zeta(\boldsymbol{x}_c) = \begin{pmatrix} r \\ \alpha \\ h \end{pmatrix} &= \begin{pmatrix} \sqrt{x^2+y^2} \\ \text{atan2}(y,x) \\ z \end{pmatrix} \\
\zeta(\boldsymbol{x}_\zeta)^{-1} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \begin{pmatrix} r\cos\alpha \\ r\sin\alpha \\ h \end{pmatrix} \\
\Theta(\boldsymbol{x}_c) = \begin{pmatrix} r \\ \phi \\ \theta \end{pmatrix} &= \begin{pmatrix} \sqrt{x^2+y^2+z^2} \\ \text{atan2}(y,x) \\ \text{acos}(\frac{z}{r}) \end{pmatrix} \\
\Theta(\boldsymbol{x}_\Theta)^{-1} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \begin{pmatrix} r\cos\phi\sin\theta \\ r\sin\phi\sin\theta \\ r\cos\theta \end{pmatrix}
\end{aligned} \tag{3.12}$$

Here, ζ is a mapping to cylindrical coordinates, while Θ maps to spherical coordinates. The components r, α, h are radius, angle, elevation and the components r, θ, ϕ are radius, altitude angle, azimuthal angle. The camera described in Section 8.4 outputs images that are already projected onto the lateral surface of a

[4] This is actually true for any omnidirectional camera system, including fish-eye lenses.



(a) Cylindrical camera model **(b)** Equiangular camera model

Figure 3.5: Illustration of the cylindrical and equiangular camera model. While the equiangular model is parameterless, the cylindrical model is described by r and h_0. In both cases, a cross-section of the image plane is highlighted in red and the path of light in orange. Further, the camera up-vector is written as $\hat{\boldsymbol{y}}$ and the vertical image bounds are marked as v_{min} and v_{max}.

cylinder. Such an image can be generated by sampling the surface in cylindrical coordinates. These coordinates are then mapped to Euclidean ones, which are subsequently mapped to image coordinates using the camera model. More details are available in [PN97] and [BPA03]. Treating this projection process as a black-box[5], a camera model representing the projection to the lateral cylinder surface has to be used, as explained in the following section.

3.2.5 Cylindrical and Equiangular Camera Models

A common application of cylindrical camera models is the representation of rotating cameras [HKS08, p.34-43]. These models are designed for non-central camera systems though, and a more compact representation is available for single viewpoint configurations. Smadja et al. discuss a central cylindrical camera model in [SBD04], which is – for simplicity – subsequently referred to as *the* cylindrical camera model.

In the vertical dimension, the cylinder model behaves like the pinhole camera model, where r translates to f and h_0 translates to u_0 or v_0. The horizontal dimension

[5]The manufacturer of the camera described in Section 8.4 states: "We have our own proprietary calibration method. However, the engineer said we are closer to Scaramuzza for the calibration and we project the image onto a cylindrical surface."[Spe15]

on the other side is parameterless[6], as the relative position of world coordinates fully describes the angle of entry. Given r and h_0, and using the notation of the previous section, the mapping from 3D to image space is straightforward. Let $\boldsymbol{x}_c$ be an arbitrary point in 3D and $(r_x, \alpha_x, h_x)^\top = \zeta(\boldsymbol{x}_c)$ be the corresponding cylindrical coordinate. Then the projection to the lateral cylinder surface is written as $(r, \alpha_x, h_0 - rh_x/r_x)^\top$. This process is illustrated in Figure 3.5a. As the surface also substitutes the image plane, it follows that $(u, v)^\top = (s_x\alpha_x, h_0 - rh_x/r_x)^\top$, where $s_x = width/2\pi$ scales from radians to pixels due to image discretization. Similarly, and as with the pinhole camera model, the parameter r is also scaled to express discretization.

Another virtual camera model is the equiangular camera model[7], which is similar to the cylindrical model and was also used during experiments. Here, not only the horizontal component is purely angle based and parameterless, as with the cylindrical one, but also the vertical component. As a result, the equiangular model is completely parameterless and angular differences are proportional to image space pixel differences. It derives analogously to the cylindrical model: When $(r_x, \phi_x, \theta_x)^\top = \Theta(\boldsymbol{x}_c)$, then $(u, v)^\top = (s_x\phi_x, s_y\theta_x)^\top$. Formulas for back-projection are given in Equation 3.12.

[6]The trivial parameters of image width and height are neglected here, since they are not subject to calibration.

[7]It is possible to construct equiangular catadioptric cameras, as described in [Cor11, p.272]. These cameras are non-central, however, and during experiments a purely virtual equiangular camera has been used to create synthetic image sequences.

Chapter 4

Multi-View Geometry

Many of the mathematical principles required for visual SLAM belong to the scope of multi-view geometry. The purpose of multi-view geometry is to accumulate the information of several camera views to obtain a meaningful description of the environment. In particular, the estimation of depth is only possible when at least two views are available, as long as the camera is the only sensor. Since multi-view geometry is commonly applied to a pinhole camera model, it is necessary to transfer relevant concepts to arbitrary central camera models. The remainder of this section provides such a transfer and shows which methods have to be adapted.

4.1 Epipolar Geometry

For the moment, assume that two images (I and I') of the same object are shot with a pinhole camera. The subject of epipolar geometry is to describe the geometrical relationship between the object and the two images. It is possible to associate a point in image I to a line in I' and vice versa, given the camera transformation. This means that a point $\boldsymbol{u}$, which is expressed in normalized homogeneous image coordinates in I, will be found on a line $\boldsymbol{l}'$ in I'. Further, the family of such lines, which are called the *epipolar lines*, meet in a single point referred to as the *epipole*. The above-mentioned relationship is called the *epipolar constraint* and a vast variety of algorithms make use of it. There are two special cases one has to consider: The point to line relationship is only valid when there is a non-zero baseline $\boldsymbol{t}$ between the first camera center C and the second camera center C'. Further, the line $\boldsymbol{l}'$ degenerates to a point in the case that $\boldsymbol{u}$ is placed at the epipole. Also, it is possible that an epipolar line is outside of the image window. A more detailed explanation can be found in textbooks, such as [HZ04]

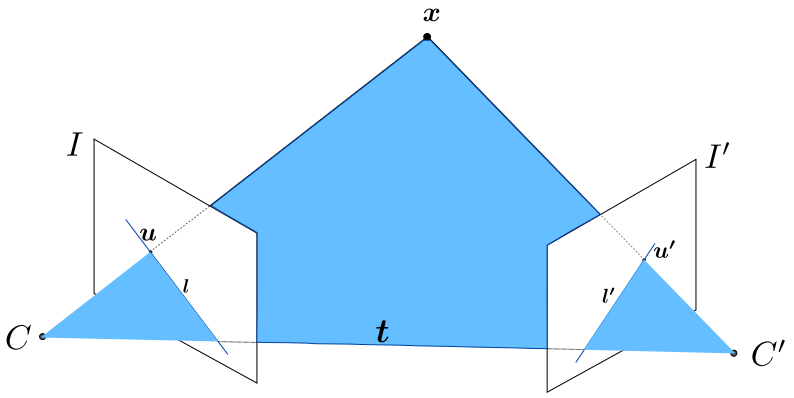


Figure 4.1: Illustration of the epipolar constraint. C, C' and $\boldsymbol{x}$ form the epipolar plane. When projecting the ray through C and $\boldsymbol{x}$ onto the image plane I' one gains the epipolar line $\boldsymbol{l}'$ which is defined by the intersection of the epipolar plane with the image plane I'.

or [FP02]. Figure 4.1 visualizes the epipolar constraint. In algebraic terms the epipolar constraint can be derived by forcing $\boldsymbol{u}$, $\boldsymbol{u}'$ and $\boldsymbol{t}$ to be coplanar:

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{t} \times \boldsymbol{R}\boldsymbol{u}') &= 0 \\ \boldsymbol{u}^\top [\boldsymbol{t}]_\times \boldsymbol{R}\boldsymbol{u}' &= 0 \\ \boldsymbol{u}^\top \boldsymbol{E}\boldsymbol{u}' &= 0 \end{aligned} \tag{4.1}$$

Where $\boldsymbol{R}$ is an orthonormal matrix rotating from reference frame C' to C and $\boldsymbol{t} = C' - C$. $\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R}$ is called the *essential matrix* that encapsulates the epipolar constraint. The plane spanned by C, C' and $\boldsymbol{x}$ is referred to as the *epipolar plane*.

A geometric interpretation of the essential matrix would be that it maps a point observation to the normal of this plane. By examining Equation 4.1 or Figure 4.2, one can see that the epipolar constraint does not only apply to projective space $\mathbb{P}^2$, but also to Euclidean space $\mathbb{R}^3$ and, thus, can be used with every camera that has a single (effective) viewpoint. One advantage of the pinhole camera model is, however, that the mapping between homogeneous coordinates and pixel coordinates is trivial. Such a direct mapping is not possible for the cylindrical camera model or omnidirectional camera models in general. Hence, a intermediate mapping step between pixel and 3D Euclidean camera coordinates is required in order to use the epipolar constraint in image space.

McMillan and Bishop introduced this mapping in 1995 [MB95] and observed that cylindrical projections to image space do not preserve lines. More specifically, the

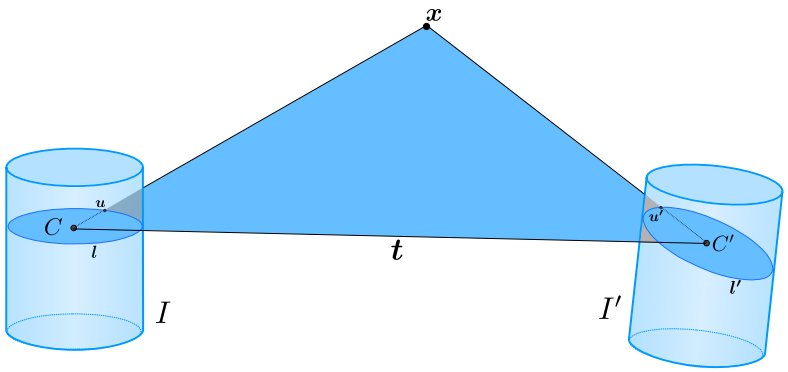


Figure 4.2: Illustration of the epipolar constraint for a cylindrical camera model.

intersections of (epipolar) planes with a cylinder are mapped to sinusoids in image space. This can be derived as follows:

$$\begin{aligned} \boldsymbol{n} &= \boldsymbol{t} \times \boldsymbol{R}\boldsymbol{u}' \\ \boldsymbol{n}\boldsymbol{u} &= 0 \\ n_x \sin\alpha + n_y h + n_z \cos\alpha &= 0 \\ h &= -\frac{1}{n_y}(n_x \sin\alpha - n_z \cos\alpha) \end{aligned} \tag{4.2}$$

Where $\boldsymbol{n}$ is the normal of the epipolar plane and $\boldsymbol{u}$ is expressed in cylindrical coordinates, as shown in Equation 3.12. Substituting this to obtain image coordinates leads to:

$$y = y_0 - \frac{r}{n_y}(n_x \sin\alpha - n_z \cos\alpha) \tag{4.3}$$

With this equation it is simple to draw the epipolar line on the camera image by iterating over $x = s_x\alpha_x$, where the parameters are the same as described in Section 3.2.5. An equivalent mapping to omnidirectional images as shown in Figure 1.1b was proposed by Svoboda et al. [SP02, SPH98]. It is shown that in the presence of central catadioptric cameras epipolar lines are mapped to conics in image space.

4.1.1 Essential Matrix Estimation

The computation of the $\boldsymbol{E}$-matrix can be performed similarly to the more common computation of the $\boldsymbol{F}$-matrix[1]. Assuming that a set of corresponding points

[1] $\boldsymbol{F} = \boldsymbol{K}^{-T}\boldsymbol{E}\boldsymbol{K}^{-1}$ is a generalization of the $\boldsymbol{E}$-matrix and incorporates camera parameters. It can be used to operate on homogeneous image coordinates, when a pinhole-camera is present. It is not used in the remaining implementation and is only mentioned here to put the method into context.

$\{(\boldsymbol{u}_1, \boldsymbol{u}'_1), ..., (\boldsymbol{u}_8, \boldsymbol{u}'_8)\}$ is known, by matching salient image features, for instance, then Equation 4.1 yields one (in the elements of $\boldsymbol{E}$ linear) equation for each correspondence. Rearranging the matrix elements of $\boldsymbol{E}$ in a column vector $\boldsymbol{e}$, the linear system has the following form:

$$\boldsymbol{u}^\top \boldsymbol{E} \boldsymbol{u}' = \boldsymbol{A}\boldsymbol{e} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1z_1 & y'_1x_1 & y'_1y_1 & y'_1z_1 & z'_1x_1 & z'_1y_1 & z'_1z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \boldsymbol{e} = 0 \tag{4.4}$$

For comparison, performing the same multiplications with the $\boldsymbol{F}$-matrix leads to:

$$\boldsymbol{u}^\top \boldsymbol{F} \boldsymbol{u}' = \boldsymbol{A}\boldsymbol{f} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \boldsymbol{f} = 0 \tag{4.5}$$

Being a 3×3-matrix, $\boldsymbol{E}$ has 9 entries, but as the aforementioned system of equations is homogeneous, 8 points are sufficient for solving it. Because $\boldsymbol{E}$ represents a rotation with 3 degrees of freedom and a translation with 3 degrees of freedom, but is only defined up to scale, it has 5 degrees of freedom in total – less than the $\boldsymbol{F}$-matrix. There are methods, like the one proposed by Li and Hartley [LH06] that exploit this property and solve the $\boldsymbol{E}$-matrix using 5 correspondences only. The implementation discussed later, however, uses a slightly modified version of the *8-point algorithm*, since the 8-point algorithm has been tested widely and is used in state-of-the-art SLAM methods, such as ORB-SLAM [RRKB11]. An in-depth discourse on the algorithm can be found in [HZ04, p.279-283]. The adjusted implementation, which is independent of the camera model, proceeds as follows:

1. For each point observation $\boldsymbol{u}_1, ..., \boldsymbol{u}'_8$ compute the normalized back-projected vector $\hat{\boldsymbol{u}}_1, ..., \hat{\boldsymbol{u}}'_8$ using the camera model.
2. Set up a linear homogeneous equation system using Equation 4.4 and at least 8 point correspondences.
3. Compute the singular value decomposition of the system. The singular vector with the smallest singular value represents an unconstrained $\boldsymbol{E}$.
4. Enforce constraints by computing another singular value decomposition of $\boldsymbol{E}$, setting the first two singular values to be equal and the last one to be 0.

A similar approach is described by Bunschoten and Kröse in [BK03] and Kawanishi et al. [KYK11]. Hartley and Zisserman explain the process of extracting camera motion, given the essential matrix in [HZ04, p.257-259].

4.2 Triangulation

In the domain of computer vision[2], 3D reconstruction refers to the process of inferring three dimensional shape information from a set of two dimensional images. Assume that a point $\boldsymbol{x}$ is given in the world coordinate system, which can be observed in two images with the coordinate $\boldsymbol{u}$ and $\boldsymbol{u}'$ respectively. When the two images (I, I') belong to two camera views with known camera parameters, then the objective to calculate $\boldsymbol{x}$ from $\boldsymbol{u}$ and $\boldsymbol{u}'$ is called the triangulation problem. Even though this sounds very feasible at first glance, it is not trivial to calculate a good estimate when dealing with different sources of noise. After all, one cannot assume that the back-projections of $\boldsymbol{u}$ and $\boldsymbol{u}'$do intersect and it has been shown that different solutions vary in precision, as described by Hartley et al. [HS97]. Researchers have been addressing this problem for more than two decades and are still doing so, due to the importance of the problem. Modern approaches try to relax the precision requirements of camera parameters and allow the incorporation of an arbitrary number of views while keeping the computational cost low. Such an approach is proposed by Recker et al. [RHFJ13] and further examined by Hess-Flores et al. [HFRJ14].

The most popular triangulation approach is *linear triangulation*, as described by Hartley et al. in [HGC92]. Even though it is well known that this method is rather imprecise, i.e. it does not minimize the geometric error and is not projective invariant, it is commonly used due to its speed and simplicity. State-of-the-art visual SLAM systems such as ORB-SLAM by Mur-Artal et al. [MAMT15] rely on linear triangulation and on the fact that successive optimization methods will compensate for the error. The principles of linear triangulation are simple: Given a point $\boldsymbol{x}$ in homogeneous world coordinates, which is regarded as projective point $\boldsymbol{u} = d(u, v, 1)^\top$, and $\boldsymbol{u}' = d'(u', v', 1)^\top$ in two camera frames so that $\boldsymbol{u} = \boldsymbol{P}\boldsymbol{x}$ and $\boldsymbol{u}' = \boldsymbol{P}'\boldsymbol{x}$ for known affine camera transformations $\boldsymbol{P}$ and $\boldsymbol{P}'$; then by performing the matrix multiplications and by doing few rearrangements, one gets an equation system of the form $\boldsymbol{A}\boldsymbol{x} = 0$. There are different methods for solving homogeneous equation systems, for example the method of Lagrange or linear least squares. As the solution of a homogeneous equation system is only defined up to scale, the solution has to fulfill an extra condition like forcing the last component to be 1, which is already required here. Since the definition of $\boldsymbol{u}$ and $\boldsymbol{u}'$ presume a pinhole camera model, this method is not directly applicable to the camera models used later. Instead, one could define $\boldsymbol{x}_c = d\frac{\boldsymbol{x}_c}{\|\boldsymbol{x}_c\|} = \boldsymbol{P}\boldsymbol{x}$ and $\boldsymbol{x'}_c$ respectively, where $\boldsymbol{x}_c$ is a normal vector pointing towards $\boldsymbol{x}$ in Euclidean camera coordinates and can be calculated using the camera model. This could also be rearranged to a homogeneous system of the form $\boldsymbol{A}\boldsymbol{y} = 0$, where $\boldsymbol{y}$ is $\boldsymbol{x}$ stacked on top of $(-d, -d', 1)^\top$ and

[2]Please note that triangulation refers to a different process in computer graphics.

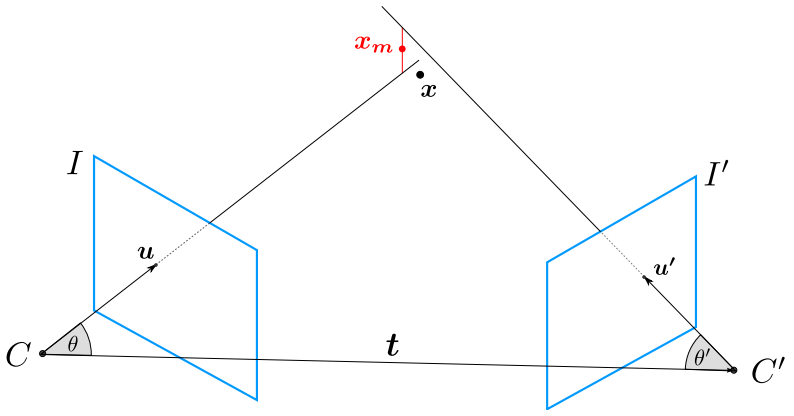


Figure 4.3: Illustration of the triangulation problem. Here, $\boldsymbol{x}_m$ denotes the solution of midpoint triangulation. By assuming that the back-projections of $\boldsymbol{u}$ and $\boldsymbol{u}'$ coincide with two edges of the triangle given by C, C' and $\boldsymbol{x}$, the length of the respective edge can be computed using θ and θ'.

$\boldsymbol{A}$ is a 6×6-matrix. The quality of such an estimation is questionable, however, because the distances d and d' of $\boldsymbol{x}$ to the corresponding camera center are not constrained to $\boldsymbol{x}$ itself. Another way to use linear triangulation with a cylindrical camera model would be to assume a local pinhole camera model by modifying $\boldsymbol{P}$ and $\boldsymbol{P}'$ accordingly. But then the error would be strongly related to the distance of $\boldsymbol{x}$ in camera coordinates to the xz-plane of the camera frame.

The *midpoint method* presents another way of solving the triangulation problem and has also been frequently suggested in literature, as in [FP02]. It supposes that $\boldsymbol{x}$ is located in the middle of the two back-projections $\boldsymbol{u}$ and $\boldsymbol{u}'$. By exploiting that the shortest connection between two rays is perpendicular to both of them, the midpoint $\boldsymbol{x}_m$ can be calculated easily:

$$\begin{aligned} \hat{\boldsymbol{n}} &= \hat{\boldsymbol{x}}_c \times \boldsymbol{R}\hat{\boldsymbol{x}}'_c \\ C + d\hat{\boldsymbol{x}}_c + s\hat{\boldsymbol{n}} &= C' + d'\boldsymbol{R}\hat{\boldsymbol{x}}'_c \\ \Rightarrow \boldsymbol{t} &= d\hat{\boldsymbol{x}}_c + s\hat{\boldsymbol{n}} - d'\boldsymbol{R}\hat{\boldsymbol{x}}'_c \\ \Rightarrow \boldsymbol{x}_m &= \frac{1}{2}(d\hat{\boldsymbol{x}}_c + \boldsymbol{t} + d'\boldsymbol{R}\hat{\boldsymbol{x}}'_c) \end{aligned} \tag{4.6}$$

Where C' and C are the camera centres and $\boldsymbol{R}$ is the rotation matrix from C' to C, while $\boldsymbol{t}$ is the translation. The second line of Equation 4.6 provides three linear equations with three unknowns. After solving this system, the last line of the equation can be used to determine the midpoint relative to C. Although Hartley et al. [HS97] have shown that the midpoint method results in inferior

precision in 1996, it is still used in the domain of omnidirectional vision because it can be applied without modification and is very fast. For example, Schönbein et al. [SRL13] used the midpoint method in 2014 to perform trifocal omnidirectional computer vision.

In 2001, Bunschoten et al. [BK01b] presented a technique for range estimation from a pair of omnidirectional images. They also used a cylindrical camera representation and suggested computing the depth values d and d' by examining the triangle formed by C, C' and $\boldsymbol{x}$. After computing the angles θ and θ', as illustrated in 4.3, they used the rule of sine to compute $d = \|C' - C\| \cdot \sin(\theta) \cdot (sin(\theta - \theta'))^{-1}$ and $d' = \|C' - C\| \cdot \sin(\theta') \cdot (sin(\theta - \theta'))^{-1}$ based on angular disparity. This procedure does not minimize any error criterion, however, it is sensitive to inaccuracies and is more expensive than the midpoint method. Schönbein and Geiger use the same technique to obtain disparity maps in a setup with two omnidirectional cameras, in [SG14].

In order to produce minimal triangulation errors, Hartley et al. [HS97] formulated the minimization criterion:

$$\begin{aligned} &minimize && e(\boldsymbol{u}, \bar{\boldsymbol{u}})^2 + e(\boldsymbol{u}', \bar{\boldsymbol{u}}')^2 \\ &subject\ to && \bar{\boldsymbol{u}} F \bar{\boldsymbol{u}}' = 0 \end{aligned} \tag{4.7}$$

Where $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{u}}'$ are points as close as possible to $\boldsymbol{u}$ and $\boldsymbol{u}'$ while fulfilling the epipolar constraint, $e(*, *)$ measures the Euclidean distance and F is the fundamental matrix. They also offered a polynomial method to solve this optimization, known as *optimal triangulation*. Kanatani et al. independently proposed the same idea in [KSN08], but designed an iterative linear method to optimize the same error function. Torr and Zissermann [TZ97] observed that both methods agree to 3 or 4 significant figures while Kanatani's method is considerably faster.

Lindstrom [Lin10] also remarks that the polynomial method is expensive to calculate, may suffer numerical stability and is non-trivial to implement. He developed an iterative method based on the work of Kanatani et al., which converges faster. Since his method only requires two iterations in most cases, he also provides a non-iterative version of his algorithm. When Lindstrom compared the computational time of his algorithm to optimal, linear and midpoint triangulation, he discovered that it performed around 50 times faster than optimal triangulation, 45 times faster than linear triangulation and 3 times faster than midpoint triangulation.

Lindstrom's method has two disadvantages that are of interest here. First, it is explicitly designed to work with two views and does not scale to three or more views. Hence, to minimize the error of a point that is present in multiple images, additional optimization techniques (commonly bundle-adjustment) are required. Secondly, the algorithm proposed by Lindstrom works with the pinhole camera model. Even

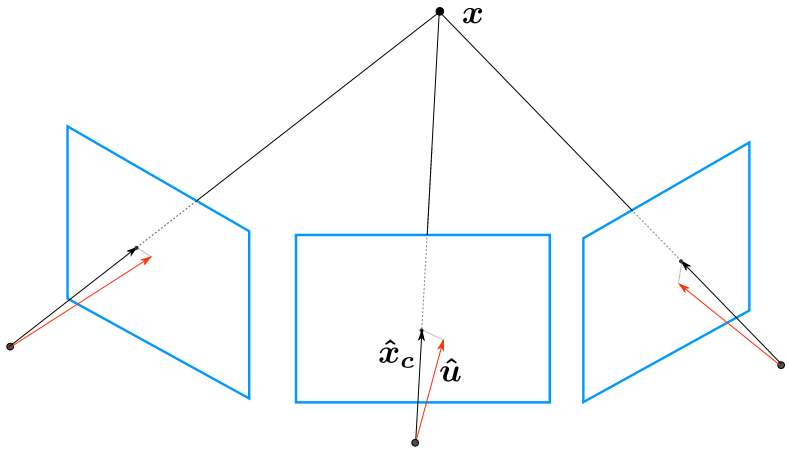


Figure 4.4: The multi-view triangulation by Recker et al. minimizes the angular error between re-projected image coordinates (red) and a scene point (black) by iteratively adjusting $\boldsymbol{x}$.

though it is possible to adapt the method for arbitrary single viewpoint models, the accuracy and performance analysis of Lindstrom becomes invalid. The same arguments are valid for the method of Nordberg [Nor08].

Another group of triangulation methods tries to minimize the error for more than two views. While there are closed-form optimal solutions for the three view triangulation case, like the method of Stewenius et al. [SSN05] and Byröd et al. [BJr07], such a method does not exist for the N-view case. Further, it is known that these methods are computationally expensive, as explained by Kukelova et al. in [KPB13] and are therefore impractical especially in the real-time domain. These problems have been addressed by Recker et al. [RHFJ13] by introducing a non-optimal *angular-based* cost function. Instead of working in the image space, Recker et al. back-project each observation of a point and try to find the 3D-coordinate that results in the lowest angular error with each back-projected ray. This approach is visualized in Figure 4.4, where red arrows correspond to reprojected point observations labeled $\boldsymbol{u}$, and $\boldsymbol{x}$ is a point whose error is evaluated. There are black arrows, one of them is labeled $\hat{\boldsymbol{x}}_c$, pointing towards $\boldsymbol{x}$ in each camera frame. The error function subject to minimization sums up the angular differences between black and red directions, using the dot-product as similarity function:

$$e(\boldsymbol{x}) = \frac{1}{\|I\|} \sum_{i \in I} (1 - \hat{\boldsymbol{x}}_{c_i} \circ \hat{\boldsymbol{u}}_i) \tag{4.8}$$

Here $\hat{\boldsymbol{u}}_i$ denotes the re-projection of the image coordinate $\boldsymbol{u}$ of the i-th camera frame. On the first glance, $e(\boldsymbol{x})$ might seem to be linear, which is not true since

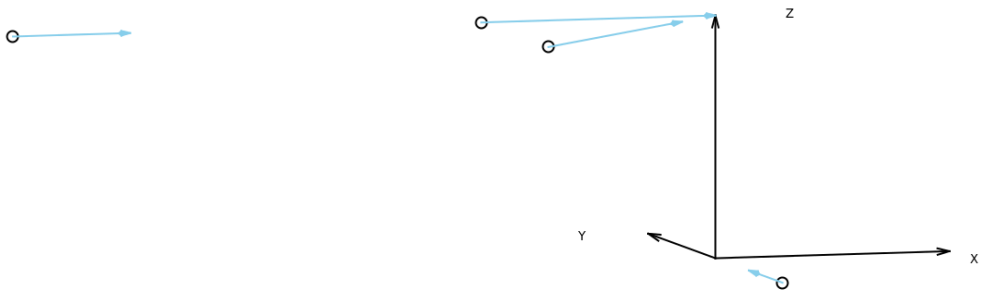


Figure 4.5: Illustration of the camera positions used in the examples. The line between $T_1 = (0,0,0)^\top$ and $T_2 = (0,0,1)^\top$ always constructs the shortest distance between the back-projected rays that are highlighted blue.

$\hat{\boldsymbol{x}}_{c_i}$ has to be normalized in each step, introducing non-linear terms. In order to minimize $e(\boldsymbol{x})$, Recker et al. suggest using gradient descent. The advantages of this approach are that an arbitrary number of camera frames can be used for triangulation and that every single-view camera model is supported. A disadvantage is, however, that Equation 4.8 does not formulate an optimal error criterion, because it discards image space information. The traditional optimality criterion of Equation 4.7 on the other hand implies that point observations in the image space are normally distributed around the "true" pixel location. Without this implication, i.e. ignoring image space completely, the best cost function would be of the form $e(\hat{\boldsymbol{x}}_{c_1}, \hat{\boldsymbol{x}}_{c_2}) = \angle(\hat{\boldsymbol{x}}_{c_1}, \hat{\boldsymbol{u}}_1)^2 + \angle(\hat{\boldsymbol{x}}_{c_2}, \hat{\boldsymbol{u}}_2)^2$, which is very similar to Equation 4.8, also expressing that the directions of the re-projections should be altered as little as possible. In this sense, the angular triangulation method shares commonalities with the principle behind bundle-adjustment, while optimal triangulation shares the exact same idea – fixed to two views though. Bundle-adjustment is covered in Section 5.3.1.

To conclude this section, the next examples illustrate the behavior of some of the methods discussed earlier. In each of them, it is assumed that two cameras observe the same point and that the shortest distance $d = 1$ between the re-projected skew rays is given by the line connecting $T_1 = (0,0,0)^\top$ and $T_2 = (0,0,1)^\top$, where T_1 projects to the point observation in frame 1 and T_2 to frame 2 respectively. Note that the scale is arbitrary here and if the viewpoint of the first camera is supposed to be $C_1 = (0,-1,0)^\top$, then the only remaining variable is the position of the second camera in the $z = 1$ plane: $C_2 = (x_2, y_2, 1)^\top$. See Figure 4.5 for an illustration of the camera configurations.

Example 1: In the initial example, the position of the second viewpoint is chosen to be $C_2 = (-1, 0, 1)^\top$. The contours of the optimal cost function of Hartley et al. and the contours of the cost function of Recker et al. in this rather extreme example – the back-projected rays are oriented perpendicular to one another – are compared in the first column of Figure 4.6. Both functions have their minimum at $(0.25, 0.25, 0.5)^\top$ surrounded by a smooth cost increase. The midpoint method would suggest to use the point $(0, 0, 0.5)^\top$ for triangulation. Triangulating the points directly, as done by Bunschoten et al. in [BK01b], would yield two possible results, one on each of the back-projected rays: $(0.5, 0, 1)^\top$ and $(0, 0.5, 0)^\top$.

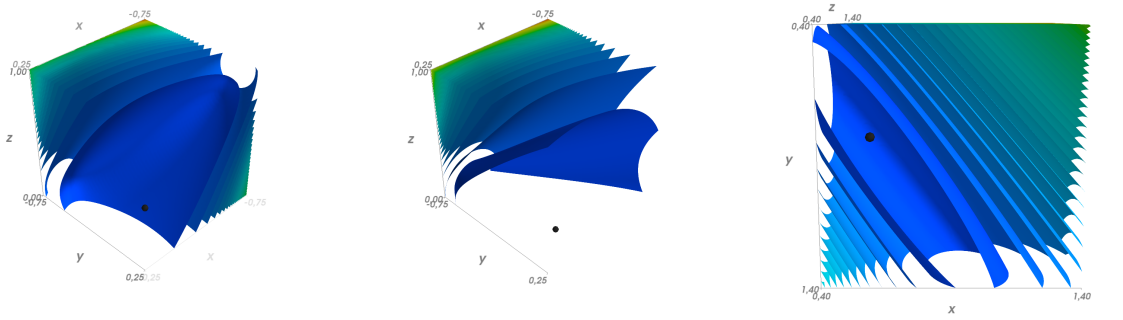
Example 2: Now, assume C_2 is located at $(-3, 0, 1)^\top$. This means the back-projected rays are still perpendicular with the same distance, but with one camera center shifted further away from the origin. At this configuration the naïvety of the midpoint method becomes obvious, which still suggests coordinate $(0, 0, 0.5)^\top$. Although C_1 is nearer at the presumable location of the 3D-point, C_2 has as much influence on the triangulation as C_1. Imagine C_2 was at $(-\infty, 0, 1)^\top$, then the first camera solely would compensate the error and the triangulated point would still be at $(0, 0.5, 0)^\top$. The results of angular and optimal triangulation are still conformable, as illustrated in the second column of Figure 4.6. Here, direct triangulation yields: $(0.191, 0, 1)^\top$ and $(0, 0.427, 0)^\top$.

Example 3: In this last example, the angle between back-projected rays is 45° with $C_2 = (-1, -1, 1)^\top$. Under these conditions, the difference between the angular and optimal cost function becomes relatively large, as the last column of Figure 4.6 shows. Yet, compared to direct triangulation, which yields $(1, 1, 1)^\top$ and $(0, 1.449, 0)^\top$, or midpoint triangulation – which did not change in this example either, since the properties of the perpendicular connecting line did not change – the results are promising. Additionally, the configurations chosen here exhibit a high perpendicular offset to the baseline ratio in order to emphasize the differences of the methods. More realistic data will result in a better performance of all methods, as long as the baseline is large enough.

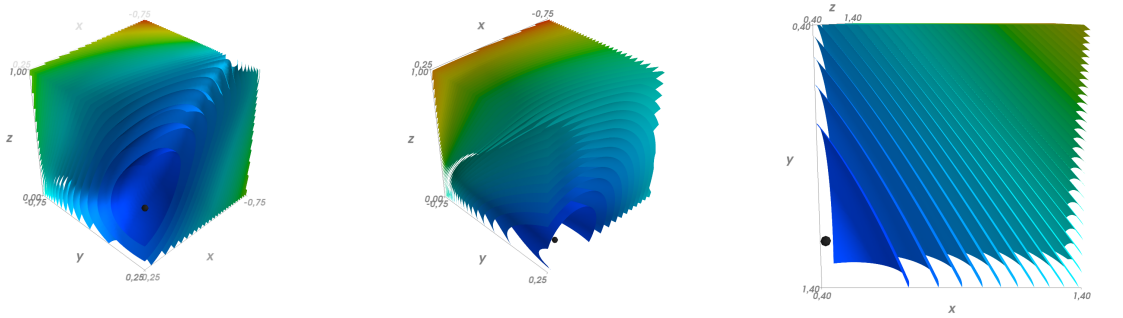
For each method, the absolute error in all three examples is listed in Table 4.1. Even though only a sparse set of examples was evaluated here, it became clear that angular triangulation performs better than midpoint and direct triangulation.

	Angular Triangulation	Midpoint Triangulation	Direct Triangulation
Ex. 1	**0**	0.354	0.612 / 0.612
Ex. 2	**0.133**	0.399	0.903 / 0.348
Ex. 3	**0.44**	1.024	0.619 / 1.017

Table 4.1: Absolute error of triangulation methods when compared to optimal triangulation. While angular triangulation performed best in all three examples, and direct triangulation performed consistently bad, in example 3 midpoint and direct triangulation performed similarly bad.



Optimal cost function at 90° $(0.25, 0.25, 0.5)^\top$ | Optimal cost function at 90° $(0.034, 0.101, 0.116)^\top$ | Optimal cost function at 45° $(0.577, 0.842, 0.577)^\top$



Angular cost function at 90° $(0.25, 0.25, 0.5)^\top$ | Angular cost function at 90° $(0.031, 0.093, 0.014)^\top$ | Angular cost function at 45° $(0.417, 1.219, 0.417)^\top$

Figure 4.6: Contours of the optimal cost function of Hartley et al. compared to contours of the angular cost function of Recker et al. The color coding ranges from blue (low error) over green to red (high error). The optimum is marked by a black dot. The volume of all six plotted squares equals to 1, and the first two columns and the last column show exactly the same range.

Chapter 5

Monocular SLAM

Monocular *structure from motion* algorithms have been being developed for more than three decades now [Har87, Lon81]. Structure from motion traditionally refers to computationally expensive methods, such as [FZ98], which evaluate a whole image sequence in order to find camera poses and world coordinates. The dominant optimization technique in this area is bundle adjustment. Even though Harris and Pike described a monocular SLAM system in 1987 [HP87], the first online monocular visual SLAM method, which performs mapping and tracking in real-time, was presented by Davison in 2003 [Dav03]. Davison's method, later called MonoSlam [DRMS07], is based on EKF-SLAM [WB95]. The system was limited in map size and required noticeable user interaction.

Both of the aforementioned drawbacks were overcome by Klein and Murray in their seminal PTAM (**p**arallel **t**racking **a**nd **m**apping) publication [KM09]. In contrast to previous work, Klein and Murray separate the mapping and tracking task, which allows fast tracking, while the map is refined using bundle adjustment. That bundle-adjustment is not limited to the offline structure from motion domain was shown by Engels et al. [ESN06]. At its core, PTAM maintains a map of sparse 3D-points that correspond to salient features (*keypoints*) in camera images, which is used for mapping and tracking – more details are provided in Section 5.3.

An alternative approach is to reconstruct a dense 3D map and to use this for tracking, as proposed by Newcombe et al. [NLD11]. Such methods are also called *direct* or *dense methods*, since they extract 3D points for as many pixels as possible. They have the advantage that more data of the image is effectively used by the SLAM system and researchers concluded [ESC14, NLD11, FPS14] that they are more robust as well as more accurate than keypoint-based methods. Both techniques, which are based on keypoints or performed directly, are still subjects

of active research and the current state-of-the-art method in terms of accuracy – ORB-SLAM [MAMT15] – is a keypoint-based method[1].

5.1 Omnidirectional Monocular SLAM

Due to the popularity of central-projection cameras, most visual SLAM systems presume a pinhole camera model. Omnidirectional computer vision has also been researched intensively, but most SLAM-systems support either the pinhole or a specialized omnidirectional camera model.

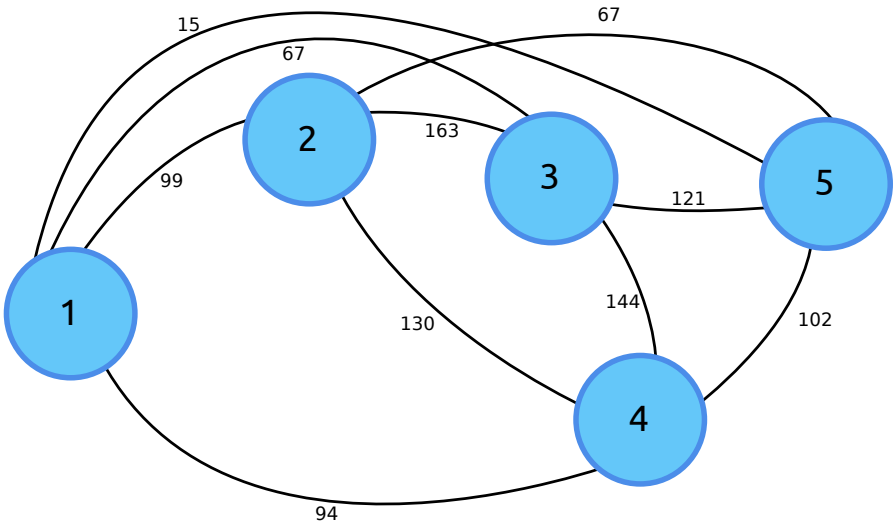
Some of the early works on omnidirectional computer vision are referred to in Section 1.1. This section continues to elaborate on the more advanced methods that have been proposed. Inspired by the work of Davison, published methods are still facilitating EKF-SLAM for omnidirectional monocular SLAM [RPG10b, RPG10a, GRMG11, PO13]. This might be due to the fact that landmarks are longer visible when an omnidirectional sensor is used. Hence, the drawback that EKF-SLAM scales badly with respect to map size is less prominent. But these methods would not be applicable for a large-scale SLAM system nevertheless. An omnidirectional monocular SLAM system based on Fast-SLAM was proposed by Gamallo et al. [GMR13] in 2013. Even though Fast-SLAM performs more efficiently than EKF-SLAM, their processes of tracking and mapping are strongly coupled. This neglects a more profound map optimization, since tracking has to be executed at framerate. With PTAM, Klein and Murray demonstrated the advantages of a decoupled mapping and tracking component.

Other methods put strong assumptions on the environment. The approach of Burbridge et al. [BSCN08], for example, only maintains a 2D map of the environment, or the work of Schoenbein and Geiger [SG14] assumes an urban Manhattan world. In [SS08], Scaramuzza and Siegwart presume planar motion.

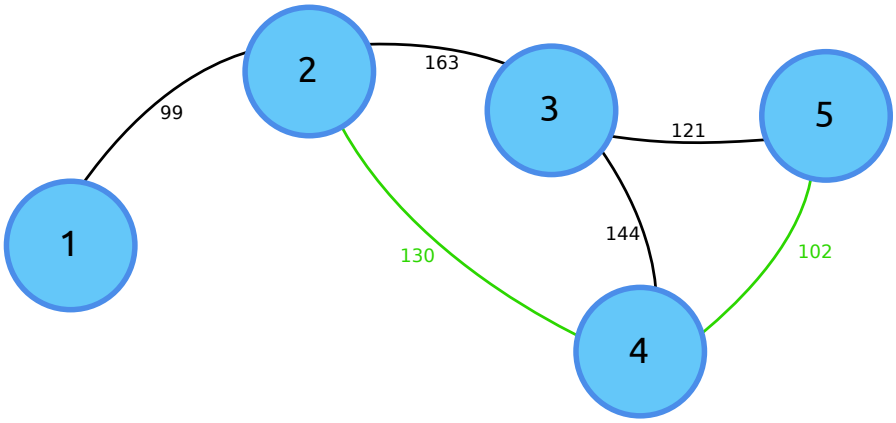
A very recent direct method that supports the unified projection model is proposed by Caruso et al. [CEC15]. It is based on LSD-SLAM and arguably the most similar one to CAM-SLAM in terms of flexibility. A strength of this omnidirectional LSD-SLAM implementation is that it handles aggressive camera motion with strong rotations well. It was presented too late to be investigated properly, though. On a first inspection, it would not support the camera models discussed in Section 3.2.5, and hence, not the V.360°.

[1] This statement does not consider the omnidirectional LSD-SLAM implementation that has just been proposed by Caruso et al. [CEC15] and might be more accurate. A superficial comparison of Table 2 of their work to Table 9.1 indicates, however, that the accuracy lies between the one of LSD- and ORB-SLAM.

5.2 Keyframe Graphs



(a) While vertices of the *covisibility graph* represent keyframes, its edges denote the number of commonly observed map points. In the presence of plentiful keyframes, the number of edges might increase considerably.



(b) Although the *essential graph* is still connected strongly, the number of edges is bound. Black edges belong to the spanning tree of the covisibility graph and green edges were kept from the covisibility graph due to the threshold (here $\lambda_c = 100$) condition.

Figure 5.1: Comparison of a covisibility and essential graph.

Direct methods and (modern) keypoint-based methods share the same overall ideas. While the camera is moving in space, a subset of tracked image frames, so-called *keyframes*, is selected and used for mapping. The very latest frame is then tracked using the created map. One might notice that this poses a

chicken-and-egg problem, which is tackled in Section 5.5. To enable global map optimization, direct methods [KSC13, ESC14], as well as keypoint-based methods [SDMK11, MAMT15], maintain a keyframe graph. Even though details differ, the following concept remains unaltered: Neighboring keyframes[2] become constrained vertices and graph optimization techniques are used for optimization. The *pose graph*, for instance, constraints keyframes by their relative poses and is frequently used for loop closing. In case a keypoint-based method is employed, neighboring keyframes could be defined to be the ones that observe the same map points. Mei et al. [MSN10] suggest constructing a *covisibility graph*, whose edges represent the number of commonly observed points. Such a graph is illustrated in Figure 5.1a. The *essential graph*, which is described by Mur-Artal [MAMT15], is the conjunction of the spanning tree of the covisibility graph with the set of edges that have a value higher than a threshold – where the spanning tree is chosen to exhibit edges with highest covisibility values. Figure 5.1b shows an essential graph. Neither the essential graph nor the covisibility graph are used for constraint optimization directly. Instead, they are used to identify neighboring keyframes and help to realize effective pose graph optimization or bundle-adjustment.

5.3 Keypoint-based Methods

A *keypoint* is a point in a camera image that exhibits special structure and that can be distinguished from other points because of its salient features. More importantly, keypoints are desired to be invariant to translation, rotation and scaling. This means that the same keypoint can be identified in two or more images, even though the camera underwent substantial motion in the meantime. At least three steps have to be performed when matching keypoints of two images. Firstly, salient image points have to be *detected* in both images. Secondly, a comparable *descriptor* for the identified locations has to be generated. The last step is to *match* the keypoints by comparing the descriptors. Well-known methods for the first two steps are SIFT [Low04] and SURF [BTVG06]. Since both methods are rather demanding with respect to processing time, more efficient alternatives like the FAST feature detector [RD06] or the BRIEF descriptor [CLSF10] have been proposed. ORB [RRKB11] uses both of the aforementioned methods to provide "an efficient alternative to SIFT or SURF". A dedicated matching technique was proposed by Muja and Lowe [ML09] and is called FLANN.

Now, assuming that a camera model is provided, the method explained in Section 4.1.1 can be applied for camera motion estimation. The required correspon-

[2]Here, the term *neighboring* does not necessarily imply that the frames are located near to each other.

dences are found using one of the keypoint detection, extraction and matching methods. In fact, one could try to design a monocular visual odometry system simply by repeating the essential matrix estimation. Since the essential matrix decomposition lacks any scale-consistency checks though, such a system would perform poorly. This is one of the reasons to create a map by triangulating keypoints after the motion was estimated: When the camera is tracked relatively to a map, the scale is theoretically in accordance with the map. Most modern monocular SLAM algorithms are inspired by PTAM and differentiate between frames and keyframes. Mapping is accomplished using keypoints that were covisible in multiple keyframes. To decide whether or not a frame should be elevated to a keyframe, most methods check for the satisfaction of a number of conditions. In PTAM, these conditions are:

1. The frame was tracked accurately
2. The latest keyframe is at least 20 frames old
3. The distance to the nearest keyframe must exceed a certain threshold[3]

The heuristic of the state-of-the-art ORB-SLAM method is similar, but additionally demands that at least 50 keypoints are tracked in the frame and, in contrast to imposing a minimum distance, a minimum visual change is enforced. This is done by requesting that less than 90% of the keypoints of the selected keyframe are visible in the current frame. A result of this modified condition is that keyframes are also created when the camera movement is mainly rotational. In such a situation ORB-SLAM could triangulate new map points using the covisibility graph, while PTAM would eventually lose tracking. Furthermore, a condition related to global relocalization is used in ORB-SLAM.

One purpose of the conditions is to restrain keyframe creation, as the mapping complexity typically is $O(N^3)$ in the number of keyframes included in optimization [ASSS10].

5.3.1 Bundle-Adjustment

The prevailing optimization approach used in the structure from motion domain and nowadays also in the scope of keypoint-based monocular SLAM is bundle-adjustment. It aims at optimizing camera and triangulated map point positions

[3]The original paper [KM09] had a typographical error, stating that the distance to the nearest keypoint must exceed a certain threshold. This can be verified when examining the function NeedNewKeyFrame(KeyFrame &kCurrent) in the **MapMaker.cc** file of the PTAM source-code, which is available at github [KM13].

by minimizing the reprojection error. It is also possible to predefine parameters (like fixing camera poses), or to estimate even more parameters (like intrinsic camera parameters) at the same time. For the two-view case, the problem could be formulated as follows:

$$\underset{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n,\boldsymbol{T}_1,\boldsymbol{T}_2}{\operatorname{argmin}} \sum_{i=1}^{n} \|\boldsymbol{u}_{i1} - \pi({\boldsymbol{T}_1}^{-1}\boldsymbol{x}_i)\|^2 + \|\boldsymbol{u}_{i2} - \pi({\boldsymbol{T}_2}^{-1}\boldsymbol{x}_i)\|^2, \tag{5.1}$$

where $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ and $\boldsymbol{T}_1, \boldsymbol{T}_2$ are the world coordinates of map points and camera frame poses, respectively. When $\boldsymbol{R}$ and $\boldsymbol{t}$ are rotation and translation of a transformation from world to camera coordinates ${\boldsymbol{T}_{1,2}}^{-1}$, then denotes ${\boldsymbol{T}_{1,2}}^{-1}\boldsymbol{x}_i = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$ a camera coordinate of $\boldsymbol{x}$. π performs the projection from camera to image coordinates and depends on the camera model at hand. Lastly, $\boldsymbol{u}_{i1}$ and $\boldsymbol{u}_{i2}$ are the observed image coordinates, belonging to the position of a salient image feature. The concept of formulation 5.1 is similar to the one of 4.7: The image space errors between the projections of the reconstructed 3D points and the observed points have to be minimized. The difference is that the epipolar constraint is neglected here and that the camera poses are optimized as well. Also, multiple points are considered simultaneously in Equation 5.1. An analysis of the structure of the bundle-adjustment problem and implementation considerations are provided by Triggs et al. in [TMHF00]. As mentioned by Jeong et al. [JNS+10], bundle-adjustment is often performed using Levenberg-Marquardt optimization, but alternatives, which are, for instance, based on Powell's dog leg non-linear least squares optimization [LAo05], are also available. The implementation discussed later represents bundle-adjustment as a graph optimization problem, as depicted in Figure 5.2. It expresses transformations using the corresponding Lie algebra.

Since map point positions and camera poses are to be optimized, they constitute the parameters and are vertices in the graph. The constraint states that 3D points projected to image space have to be close to their observed position. The cost function of each constraint is the squared absolute error in image space, as in Equation 5.1. Ultimately, the graph optimization software g^{2}o uses Levenberg-Marquardt least squares optimization to perform bundle-adjustment. Therefore, a realistic initial estimate is required in order to reach an, ideally global, optimum.

Given enough camera movement, the map generated by a visual SLAM system becomes too big to be feasible for online bundle-adjustment after a short period of time. Consequently, only a local map with limited keyframes and map points is used for mapping. An effective way to select the local map was proposed by Strasdat et al. [SDMK11]. Around the currently active keyframe, an *active window* of additional keyframes is selected based on their covisibility. In their work, they split the active window into an inner and outer window. While the inner window is subject to bundle adjustment, the outer window acts as an additional stabilizer by

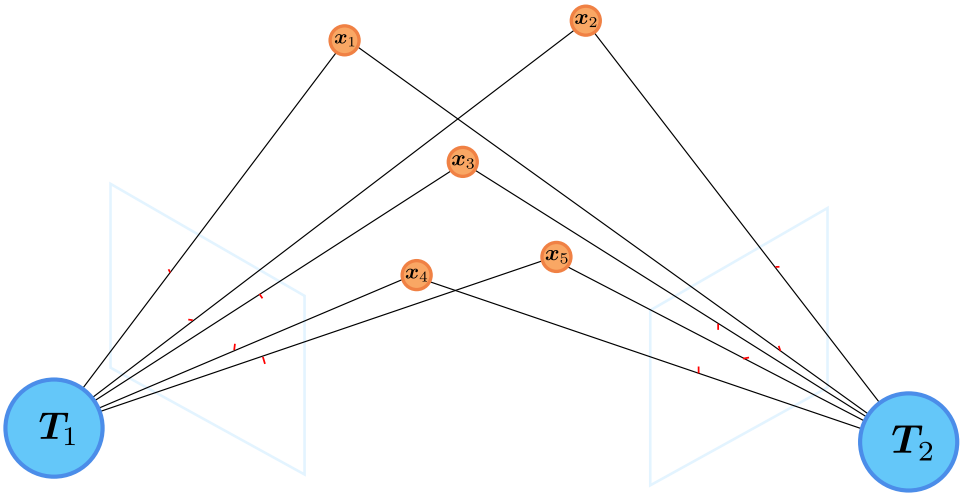


Figure 5.2: Bundle-adjustment posed as graph-optimization problem. Blue vertices represent the camera poses and orange vertices point positions that have to be optimized. Red dashes on the image planes illustrate the errors of constraints. There are point-to-camera constraints only – neither point-to-point, nor camera-to-camera constraints are part of bundle-adjustment.

introducing pose-pose constraints. Mur-Artal et al. [MAMT15] also follow a double window strategy. They include the outer window keyframes in bundle-adjustment, but fix their poses.

5.3.2 Tracking

SLAM systems in general estimate the pose of the robot based on a motion estimate, and monocular SLAM is no exception to this. While some applications [KJLS14] employ a Kalman filter in order to include higher order information, i.e. acceleration components, constant velocity motion models also proved to predict the camera pose good enough [MAMT15]. More specifically speaking, camera poses are frequently represented using the Lie group $\mathbf{SO}(3)$ or $\mathbf{Sim}(\mathbf{3})$ and relative poses are of the form $\boldsymbol{T}_1^{-1}\boldsymbol{T}_2$, where $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ are elements of the according group. Hence, considering the last two succeeding camera poses, the next pose is estimated by $\boldsymbol{T}_3 = \boldsymbol{T}_2\boldsymbol{T}_1^{-1}\boldsymbol{T}_2$.

Given the initial pose estimate, the camera position is refined using local map information. For this reason, map points of covisible keyframes are projected to the current frame and matched with nearby image features. The resulting correspondences are then used to perform bundle adjustment, optimizing the current camera position only. This approach was already proposed in the PTAM publication and more recent methods fine-tuned the implementation for their needs.

For example, the PTAM application does not maintain a covisibility graph and additional matching criteria, such as including the mean observation direction of map points, were employed by Mur-Artal et al. [MAMT15].

5.4 Direct Methods

Before the introduction of dense monocular SLAM methods, state estimation was either filtering-based (EKF-, FAST-SLAM) or based on bundle-adjustment. Newcombe et al. presented another strategy labeled ***d**ense **t**racking **a**nd **m**apping* (DTAM) [NLD11] that differs from the former ones by (potentially) involving complete images in optimization. As mentioned earlier, the overall concept is comparable to the one of PTAM, although the optimization back-end works entirely distinctly. Assuming that a dense reconstruction of the environment is already available and that a motion-based estimate of the current camera pose was predicted, the camera is tracked by iteratively minimizing the *photometric error*. The photometric error refers to the (sum of) pixel value differences – L_1-norm [NLD11], or squared L_2-norm [ESC13] – between the actual current image and the image generated by projecting the dense map to the estimated camera location. One can think of this procedure as moving a virtual camera and comparing the projected virtual camera image to the real one. Mathematically, the photometric error is of following nature:

$$e = \sum_{i=1}^{n} \| I_1(\boldsymbol{u}^i) - I(\pi(\boldsymbol{x}_c{}^i) \|^{1,2}, \tag{5.2}$$

where I_r, I maps from image coordinates to pixel intensity values and $\|\cdot\|^{1,2}$ expresses that the value is either squared or not, depending on the error-norm. The major difference from Equation 5.1 originates from comparing these intensity values rather than image space (pixel) offsets. Moreover, e is evaluated densely in contrast to the cost function of 5.1. The reasoning behind the photometric error derives from the constant brightness assumption: When the image of an object moves, it does not change brightness. Hence, e would converge to a global minimum near a perfectly tracked camera, if it wasn't for noise and if the constant brightness assumption was completely true. Since actual dense monocular SLAM systems track the current frame relatively to a selected reference keyframe and depth image, literature describes the photometric error in a slightly more elaborate fashion:

$$e = \sum_{i=1}^{n} \| I_r(\boldsymbol{u}^i) - I(\pi(\boldsymbol{T}\pi^{-1}(\boldsymbol{u}^i, d^i)) \|^{1,2} \tag{5.3}$$

Here, $\pi^{-1}(\boldsymbol{u}, d)$ denotes a back-projection of $\boldsymbol{u}$ to a 3D coordinate at depth d. The resulting 3D point is then transformed from one camera coordinate system to the other one using the relative transformation $\boldsymbol{T}$ and finally projected to image space again. Note that in contrast to the formulation of Newcombe et al. [NLD11, eq.2-3] all camera parameters are implied in π here, hence, multiplying with K would be redundant[4]. The $w(\boldsymbol{u}^i, \boldsymbol{T}) = \pi(\boldsymbol{T}\pi^{-1}(\boldsymbol{u}^i, d^i)$ component that maps from image to image space is also called *warp function*. After system initialization, all parameters except the relative transformation are available. The *Lucas-Kanade image alignment* algorithm presents a method to minimize e using nonlinear optimization. A comprehensive analysis of the algorithm and Gauss-Newton based implementation details are provided by Baker and Matthews in [BM04]. In addition to direct methods that minimize the photometric error and keypoint-based methods that minimize the re-projection error, a semi-direct method has also been proposed by Forster et al. [FPS14] and is called SVO. SVO maintains sparse features and tracks the camera by reducing the photometric error of patches around the features. Subsequently, bundle-adjustment is used to refine the tracking.

As with keypoint-based methods, a keyframe-decision is made after tracking a frame successfully. DTAM uses the number of projected pixels as a threshold. When this falls below a certain value, a keyframe is created because future tracking might become unstable otherwise. The more recent LSD-SLAM system presented by Engels et al. [ESC14] bases the threshold on a weighted combination of relative distance and angle to the currently selected keyframe. Once a new keyframe is created by a direct method, either a dense or semi-dense depth-map is generated. LSD-SLAM, for instance, employs the stereo-matching earlier presented by Engels et al. [ESC13] for this task. The depth value d in subsequent tracking activities is then read from the newly created depth-map. Each keyframe initializes its depth map based on the projected depth map of the tracking process and expands the depth map using epipolar line searches and triangulation.

5.5 Initialization

Section 5.3 already described that an essential matrix estimation based on the 8-point algorithm yields a camera motion estimate. Given a motion estimate, it is also possible to triangulate points, and hence, to initialize keypoint-based visual SLAM methods. This way, the chicken-and-egg problem due to a *tracking task depending on mapping* and a *mapping task depending on tracking* is overcome.

[4]This way the formulation is more general and fits nicely to Equation 5.1. It is equivalent to the one of Forster et al. [FPS14].

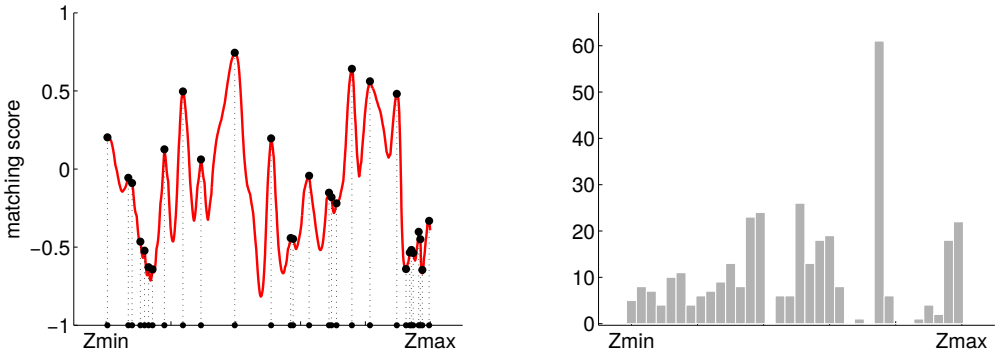
There are degeneracies associated with the 8-point algorithm, however, as explained by Hartley and Zissermann [HZ04, p.295-297]. Most importantly, applying the algorithm to points on a plane will lead to multiple solutions and causes it to fail. A homography estimation on the other hand, which also allows the reconstruction of camera motion, is only suitable for a planar scene structure. For this reason, Mur-Artal et al. [MAMT15] evaluate the fundamental matrix $\boldsymbol{F}$ as well as the homography $\boldsymbol{H}$ in parallel during initialization. A heuristic based on symmetric transfer errors is then used to decide which model is appropriate. While $\boldsymbol{F}$ can easily be replaced using the essential matrix to gain camera model independence, as in Section 4.1, it is more difficult to replace the homography estimation. Zhang et al. [ZLZH10] proposed a solution for the para-catadioptric case that can be adapted to catadioptric cameras with hyperbolic and elliptical mirrors, but a camera model independent solution demands further research.

First direct methods [NLD11, ESC13] were bootstrapped by keypoint-based methods: As soon as the keypoint-based method identifies frames with enough parallax, (semi-)dense stereo matching is performed to generate an initial map. When this procedure succeeds, the system continues to run self-supported, omitting keypoints. LSD-SLAM managed to fully exclude keypoints by starting with a random depth map and assuming a large variance in (inverse) depth. The system subsequently converges to a valid depth map, given enough translation.

Chapter 6

Semi-Dense 3D Reconstruction

Robots that are supposed to interact with the environment require a representation of their surroundings, such as the map of a visual SLAM method. Clearly, a representation as accurate as possible is desired in order to infer useful information. Keypoint-based visual SLAM could barely be used for collision detection for example, because an object nearby could lack associated image features. For this reason, a dense or semi-dense 3D reconstruction of the environment is preferable. Repeating point triangulation for as many points as possible would result in a low quality 3D scene reconstruction, due to the presence of noise and outliers. A natural way to cope with these issues is to treat reconstruction as a probabilistic process. Broadhurst et al. [BDC01] employed a Bayesian probability framework for reconstruction, which is based on space carving [KS00]. Ultimately, space carving reconstructs surfaces by analyzing the photo-consistency of voxels projected to image spaces. During this process, occlusion has to be modeled explicitly. A more recent method that models occlusion implicitly as a source of noise was presented by Vogiatzis and Hernández [VH11]. The basic principle, which is also used in other works [PNF$^+$08, ESC13, PFS14, ESC14, MAT15], is the following: Each pixel of the camera can be interpreted as a depth sensor, if the camera motion is known and when a correspondence to a pixel of a previous image is given. In this case, triangulation could be used to compute the depth. Here, a problem arises from finding the dense correspondences, which is usually done by searching the epipolar line. Another challenge is to account for noisy data and outliers. All of the aforementioned methods rely on the fusion of multiple depth measurements in order to calculate the posterior depth. Additionally, some of the methods perform a smoothing step by introducing a regularization term. After correspondences are found and depth observations are fused, the resulting (semi-)dense depth map is equivalent to a local point cloud. The 3D points of this cloud emerge from back-projecting an image coordinate to the according depth.



(a) Matching scores along an epipolar line. **(b)** Histogram of local maxima.

Figure 6.1: The plot in (a) represents the score of pixels during the epipolar line search and black dots highlight local maxima. In (b), local maxima of 60 images are accumulated into a histogram. Courtesy of George Vogiatzis, Aston University and Carlos Hernández, Google Inc. Published in [VH11] as Figure 2.

6.1 Depth Uncertainty

By interpreting the camera as a depth sensor, the following question arises: Which uncertainty is associated with each of the depth measurements?

Depending on the baseline between two camera poses, the search interval on the epipolar line might exhibit several matches to the projected pixel. This means that the epipolar search is a source of uncertainty. Given the baseline, the length of the epipolar line is determined by a depth range of interest. This depth range is either obtained by a SLAM algorithm or by prior knowledge of the scene. Hence, sampling possible depth values of a back-projected pixel resembles an epipolar search in other image frames.

At this stage, camera poses are usually assumed to be fixed and are not included in probability estimation [VH11, PFS14, MAT15]. Figure 6.1a depicts the score of potential correspondences along the epipolar line. Vogiatzis and Hernández [VH11] accumulated local maxima for the same pixel over 60 neighboring frames into a histogram and observed that it has a peak at the true depth, which is bounded by mostly uniform noise. Such a histogram is shown in Figure 6.1b. To account for this structure, Vogiatzis and Hernández represented the uncertainty by a *Gaussian + Uniform* mixture model:

$$p(z|\, z^*, \pi) = \gamma\, \mathcal{N}(z|\, z^*, \sigma^2) + (1-\gamma)\, \mathcal{U}(z|\, z_{\min}, z_{\max}) \tag{6.1}$$

The values $z_{\min}$ and $z_{\max}$ cover the scene depth, σ^2 describes the variance of a good measurement and γ the probability of an inlier observation. While the normal

distribution $\mathcal{N}$ is scaled by γ, the uniform distribution $\mathcal{U}$ is scaled by $(1-\gamma)$. This way, the probability density function $p(z|\, z^*, \gamma)$ has a peak at the true depth z^*, where the significance is based on the parameter γ. To solve Equation 6.1 for the true depth z^*, Vogiatzis and Hernández first used a Bayesian approach, which required maintaining a histogram for each epipolar line of a pixel that is observed in other frames. Because of the computational complexity of the Bayesian method, an approximation of the posterior using a *Gaussian* $\times$ *Beta* model was presented as well. This choice was motivated by the small Kullback–Leibler divergence between the actual and the *Gaussian* $\times$ *Beta* distribution. Please refer to the work of Vogiatzis and Hernández for more details on the probability estimation. Pizzoli et al. [PFS14] presented a method, which is called REMODE (**re**gularized **mo**nocular **d**epth **e**stimation), built on the results of Vogiatzis and Hernández. The key improvements of REMODE are the application of *regularization* and *inverse depth parametrization*.

Modern methods do not parametrize geometry by depth values directly, but use an inverse depth parametrization instead. This concept was presented by Civera et al. [CDM08] and offers two main advantages. First, uncertainties behave more Gaussian when using the inverse depth parametrization, which is important for probabilistic reconstruction, especially depth hypothesis fusion. Second, employing inverse depths reduces a bias during regularization: Due to projection, surfaces nearby are sampled by more pixels than distant surfaces of the same size. Hence, non-inverse depth gradients are differently scaled in fore- and background. The inverse parametrization, on the other hand, behaves linearly to a certain degree and helps to surpass this problem.

While Equation 6.1 assigns a probability density function to the sampled depth-range, it is also possible to assign similarity errors, i.e. matching costs between the back-projected pixel and pixels on the according epipolar lines. A comparison of 15 available cost functions was presented by Hirschmüller and Scharstein [HS09].

Engel et al. [ESC13] as well as Mur-Artal and Tardós [MAT15] effectively calculate errors based on grayscale and gradient images. Mur-Artal and Tardós, for instance, define the similarity error $e(u)$ of a pixel at coordinate u on an epipolar line[1] as

$$e(u) = \frac{r_I^2}{\sigma_I^2} + \frac{r_G^2}{\sigma_G^2}, \tag{6.2}$$

which is to be understood as the summation of a normalized intensity residual $r_I = I - I(u)$ and gradient residual $r_G = G - G(u)$, where $I, I(u)$ are the intensities and $G, G(u)$ the gradient magnitudes, respectively.

[1] Here, u is a one-dimensional parameter of the epipolar line, because the actual 2D-coordinate $\boldsymbol{u}$ has only one degree of freedom.

In addition to having a low similarity error, matching candidates must fulfill the following conditions:

1. The gradient must be high ($G(u) > \lambda_G$), which reduces the ambiguity of matches.

2. The orientation of the gradient $\Theta(u)$ must not be (nearly) perpendicular to the epipolar line ($|\Theta(u) \boxminus \Theta_l| < \lambda_{\Theta_l}$). This way, the ambiguity is further reduced[2], as described in-detail in [ESC13].

3. The orientation $\Theta(u)$ has to match a prediction based on the median rotation of matched ORB features. ($|\Theta(u) \boxminus \Theta_p| < \lambda_{\Theta_p}$)

Here, only the first two conditions are enforceable for every single-view camera model, since the last condition assumes evenly distributed rotations. As a result of using the Scharr operator for gradient computation, Mur-Artal and Tardós relate the intensity noise to the gradient noise by $\sigma_G^2 = \theta\sigma_I^2$, where $\theta = 0.23$. This yields the similarity error

$$e(u) = (r_I^2 + \frac{1}{\theta} r_G^2)\frac{1}{\sigma_I^2}. \tag{6.3}$$

By sampling the epipolar line, the pixel coordinate u_0 with lowest similarity error is chosen to represent the correspondence in the currently evaluated reference frame. The coordinate u_0 can further be refined to sub-pixel precision by constraining the derivative of the error to be a local extremum:

$$\begin{aligned} \frac{\partial e(u)}{\partial u} = 0 &= 2(r_I' r_I + \frac{1}{\theta} r_G' r_G)\frac{1}{\sigma_I^2} \\ 0 &= -\frac{2}{\sigma_I^2}(I' r_I + \frac{1}{\theta} G' r_G) \end{aligned} \tag{6.4}$$

Where $r_I'(u) = -I'(u) = \frac{1}{2}(I(u+1) - I(u-1))$ and $r_G'(u) = -G'(u) = \frac{1}{2}(G(u+1) - G(u-1))$.

[2]Otherwise, edges of the image could (nearly) coincide with the epipolar line, and lead to a succession of matching candidates. As a result, a high uncertainty would be associated with each candidate.

This can be solved for u^* by applying a first order Taylor approximation of the residuals r_I and r_G at position the u_0:

$$\begin{aligned}
0 &= -\frac{2}{\sigma_I^2}\left(I^{'}(u_0)r_I(u_0) + \frac{1}{\theta}G^{'}(u_0)r_G(u_0)\right) \qquad (6.5)\\
&\approx -\frac{2}{\sigma_I^2}\left(I^{'}(u_0)r_I(u_0+\triangle u) + \frac{1}{\theta}G^{'}(u_0)r_G(u_0+\triangle u)\right)\\
&= -\frac{2}{\sigma_I^2}\left(I^{'}(u_0)r_I(u_0) - \triangle u I'^2(u_0) + \frac{1}{\theta}G^{'}(u_0)r_G(u_0) - \frac{1}{\theta}\triangle u G'^2(u_0)\right)\\
&\Rightarrow \triangle u(I'^2(u_0) + \frac{1}{\theta}G'^2(u_0)) = I^{'}(u_0)r_I(u_0) + \frac{1}{\theta}G^{'}(u_0)r_G(u_0)\\
&\quad \triangle u = \frac{I^{'}(u_0)r_I(u_0) + \frac{1}{\theta}G^{'}(u_0)r_G(u_0)}{I'^2(u_0) + \frac{1}{\theta}G'^2(u_0)}
\end{aligned}$$

Then, the refined position is given through $u^* = u_0 + \triangle u$ and corresponds to an inverse depth value ρ^* at the back-projected pixel. Note that the choice of the predefined parameter σ_I^2 does not affect the mininima of $e(u)$ and in consequence neither u_0 nor u^*. Mur-Artal and Tardós propagate the uncertainty σ_I^2 to depth uncertainties, which offer useful information at the stage of depth hypothesis fusion. Employing non-linear uncertainty propagation [Ku66], the variance in u^*, denoted by $\sigma_{u^*}^2$, is approximated by:

$$\begin{aligned}
\sigma_{u^*}^2 &\approx \left|\frac{\partial u^*}{\partial r_I(u_0)}\right|^2 \sigma_I^2 + \left|\frac{\partial u^*}{\partial r_G(u_0)}\right|^2 \theta\sigma_I^2 \qquad (6.6)\\
&= \frac{r_I^{'2}(u_0)}{\left(r_I^{'2}(u_0) + \frac{1}{\theta}r_G^{'2}(u_0)\right)^2}\sigma_I^2 + \frac{\frac{1}{\theta^2}r_G^{'2}(u_0)}{\left(r_I^{'2}(u_0) + \frac{1}{\theta}r_G^{'2}(u_0)\right)^2}\theta\sigma_I^2\\
&= \frac{2\sigma_I^2}{r_I^{'2}(u_0) + \frac{1}{\theta}r_G^{'2}(u_0)} = \frac{2\sigma_I^2}{I'^2(u_0) + \frac{1}{\theta}G'^2(u_0)}
\end{aligned}$$

Let $\varrho(u)$ be a mapping from a location on the epipolar line to inverse depth. Then starting from the inverse depth $\rho = \varrho(u^*)$, the standard deviation σ_ρ is inferred from two inverse depth deviations. These deviations are based on inverse depths related to two more points on the epipolar line, which are located at $u^* + \sigma_{u^*}$ and $u^* - \sigma_{u^*}$:

$$\sigma_\rho = \max(|\varrho(u^* + \sigma_{u^*}) - \rho|, |\varrho(u^* - \sigma_{u^*}) - \rho|) \qquad (6.7)$$

So far, the inverse depth ρ and standard deviation σ_ρ of a pixel is estimated by sampling the epipolar line of one reference keyframe. To increase the precision

of the measurement, multiples of such depth hypothesis from different frames are fused. Because it is likely that some hypothesis are outliers due to mismatching, a compatible subset of the measurements has to be selected. The χ^2 test offers a method to test the compatibility of hypothesis. Testing each hypothesis with the remaining ones will result in a maximal subset of compatible hypothesis. Only if the size of the subset is larger than a threshold λ_n, a fused inverse depth hypothesis ρ_f is obtained in the following manner:

$$\rho_f = \frac{\sum \frac{1}{\sigma^2_{\rho_i}} \rho_i}{\sum \frac{1}{\sigma^2_{\rho}}} , \qquad \sigma^2_f = \frac{1}{\sum \frac{1}{\sigma^2_{\rho}}} \tag{6.8}$$

Although the fusion of several measurements reduces the noise, the output inverse depth map still suffers from spurious discontinuities. The removal of such noisy discontinuities, while preserving edges, is thoroughly studied in image processing and called *image smoothing*. Several image smoothing techniques and their mathematical relations are studied by Mrázek et al. in [MWB06]. Methods based on *total variation regularization* [PFS14] or inspired by *bilateral filtering* [ESC13, MAT15] were successfully applied to improve the quality of inverse depth maps. Engel et al. [ESC13] compute the weighted average of the surrounding of each inverse depth value to efficiently regularize the depth map. The weight is based on the variance σ^2_f of the inverse depth values, and values that differ more than $2\sigma_f$ are neglected in order to preserve edges. At this stage, it is also possible to include intensity variations in the weighting, since discontinuities in color tend to correspond to discontinuities in depth.

Chapter 7

Loop-Closing

Section 2.3.6 motivated the use of global data in SLAM systems. Even if modern direct and keypoint-based visual SLAM methods operate with high precision, a drift due to accumulated errors is inevitable on large-scale data. A key concept to address this problem is called *loop-closing*. It refers to the problem of recognizing a previously visited location (*loop-closing detection*) in order to acquire a globally consistent map and trajectory (*loop-closing correction*). Williams et al. [WCN+09] differentiate between three categories of methods performing the first task of loop-closing detection: *Map-to-map*, *image-to-image* and *image-to-map* methods. Algorithms of the first category divide the map into sub-maps and a loop is detected when sub-maps exhibit similar salient features of some kind. In contrast to map-to-map techniques, image-to-image techniques operate directly on camera frames. They compare images using a visual vocabulary, for instance based on SURF features, and identify a loop based on this comparison. The latter category of image-to-map techniques includes algorithms that search the map for features observed in the current frame. Again, when the search is successful, a loop is detected.

Even though Williams et al. emphasized image-to-map methods in their work of 2009, as they potentially use more information due to the availability of 3D map-points, it seems that the focus in recent years is on image-to-image methods. Cummins and Newman presented a seminal work on image-to-image methods [CN07] that was later extended and dubbed ***f**ast **a**ppearance-**b**ased **map**ping* (FAB-MAP) [CN08]. The success of FAB-MAP derives from three qualities. First, it scales well – namely, linearly to the number of processed images. Second, it addresses *perceptual aliasing* effectively. Perceptual aliasing describes that distinct locations may visually appear coincident to the system. Third, FAB-MAP generates a probability density function over locations, which means that not only

the most compatible frames are identified, but also that the probability of a loop-detection is determined. An open-source implementation of FAB-MAP, called OpenFABMAP, is presented by Glover et al. [GMW+12]. While the loop-closing detection of LSD-SLAM is based on this implementation, the loop-closing detection technique of ORB-SLAM, presented by Mur-Artal and Tardós [MAT14], operates in a comparable manner. Like FAB-MAP, the method also uses a bag-of-words [GLT12] representation of frames: Feature descriptors are grouped to visual words by discretization, and hence, an images holds a bag-of-words as it contains a multiset of descriptors. FAB-MAP, as well as the method of Mur-Artal and Tardós, requires a pre-calculated visual vocabulary.

As soon as a loop is detected, loop-closing correction has to be triggered. Considering that the ends of the loop potentially have greatly erroneous relative poses, the "true" relative transformation has to be estimated. Since loop-closing is usually performed among keyframes using image-to-image detection methods, both ends of the loop exhibit 3D data. Accordingly, keypoint-based methods estimate the relative camera transformation between the two involved keyframes based on 3D point correspondences, which can be found by feature matching. A closed-form algorithm for this task is presented by Horn [HHN88]. Alternatively[1], the estimate is assessable using a subset of three 3D points with the method of Arun et al. [AHB87] in a RANSAC-scheme, as outlined by Strasdat [Str12a, p.129]. The resulting relative transformation estimate can further be refined using subsequent bundle-adjustment. Engel et al. [ESC14] discovered that the tracking component of their direct method is able to compute the relative transformation, when an aggressive coarse-to-fine approach (starting at resolution 20x15) is chosen. Engel et al. also expound that at the occasion of loop-closure, direct methods could still rely on keypoints in order to execute Horn's technique for transformation estimation. Given the "true" relative transformation, the accumulated error is split over all keyframes in the loop using pose graph optimization, as in [OLT06]. In order to affect scale drifts effectively at this stage, transformations are usually represented in the Lie group $\mathbf{Sim}(3)$.

[1] According to [AHB87], the methods of Horn and Arun et al. were developed independently at the same time.

Chapter 8

Implementation

Previous chapters outlined crucial principles and presented state-of-the-art monocular visual SLAM methods. This chapter revisits introduced topics and expands on their interaction with focus on the development of a real-time monocular visual SLAM system. In contrast to most existing systems, the newly implemented application – dubbed *CAM-SLAM*, which stands for ***c**amera **a**gnostic **m**onocular SLAM* – is not restricted to pinhole cameras, but explicitly allows omnidirectional and other central camera models. In fact, flexibility is one of the main characteristics of the application and has been a major reason for developing a monocular visual SLAM system from scratch. CAM-SLAM is keypoint-based and the architecture of the system is based on the following design choices:

1. **Cameras-agnostic.** First and foremost, the support for omnidirectional camera models is required, as this is the focus of this work. As a consequence, it has been investigated whether it is possible to support a wider range of camera models. The collection of algorithms utilized in the implementation successfully enables various camera models, as long as they allow feature tracking and exhibit a single effective viewpoint.

2. **Various features.** Since history indicates that new salient image feature detectors and descriptors emerge every few years, a system that is independent of the actual detector and descriptor implementation is preferable.

3. **Multi-threaded.** Modern SLAM methods exploit the multi-tasking capabilities of today's processors. As latencies can be reduced this way, the new system has to benefit from parallel computing techniques.

4. **Portable.** Finally, the application should be easily portable. Therefore, the system avoids depending on the robot operating system ROS [QCG+09].

Instead, an optional ROS node is being developed in order to include CAM-SLAM in ROS *stacks*.

At the very beginning of the development, CAM-SLAM was supposed to become a direct method, like LSD-SLAM. Experiments showed, however, that direct methods depend on an accurately calibrated camera model, which is not available for the VSN V.360° camera. This dependency results from the vast amount of epipolar line searches at each keyframe, due to the extension of the depth map. If a badly calibrated camera disturbs this process, epipolar line searches produce mismatches and corrupt depth maps. This issue is less prominent in keypoint-based methods, as correspondences are found by feature matching. Even though an epipolar check might be used to verify good matches, this step is optional for keypoint methods. Further, as explained in Section 8.3.3, sampling the epipolar line might be considerably more expensive for omnidirectional cameras, depending on the actual model. This again favors keypoint-based methods, when the camera is not represented by the pinhole model. Another decisive factor, prompting the shift to a keypoint-based method, has been the release of ORB-SLAM. As noted in Chapter 5, ORB-SLAM is today's reference keypoint-based visual SLAM system and is the most accurate system available.

While the next section continues to describe CAM-SLAM on a broad level, implementation details follow in Section 8.3.

8.1 System overview

CAM-SLAM is divided into four packages: The core package *cam-slam*, the viewer package *cam-slam-viewer*, the testing package *cam-slam-tests* and the ROS node package *cam-slam-ros*, which is still in development and not part of this thesis. The former packages are described in the following.

8.1.1 Package: cam-slam

Mandatory dependencies:	Eigen3 [GJ+10], g²o [GKSK11], OpenCV [Bra00], Sophus [Str12b], Boost (Multi-index Containers [Mun04]), C++14
Optional dependencies:	OpenSeqSLAM [Sue13], OpenML [DM98], ORB-Extractor [MA15] of ORB-SLAM, CamOdoCal [HLP13]

The cam-slam core component abstracts all methods from the user and provides an

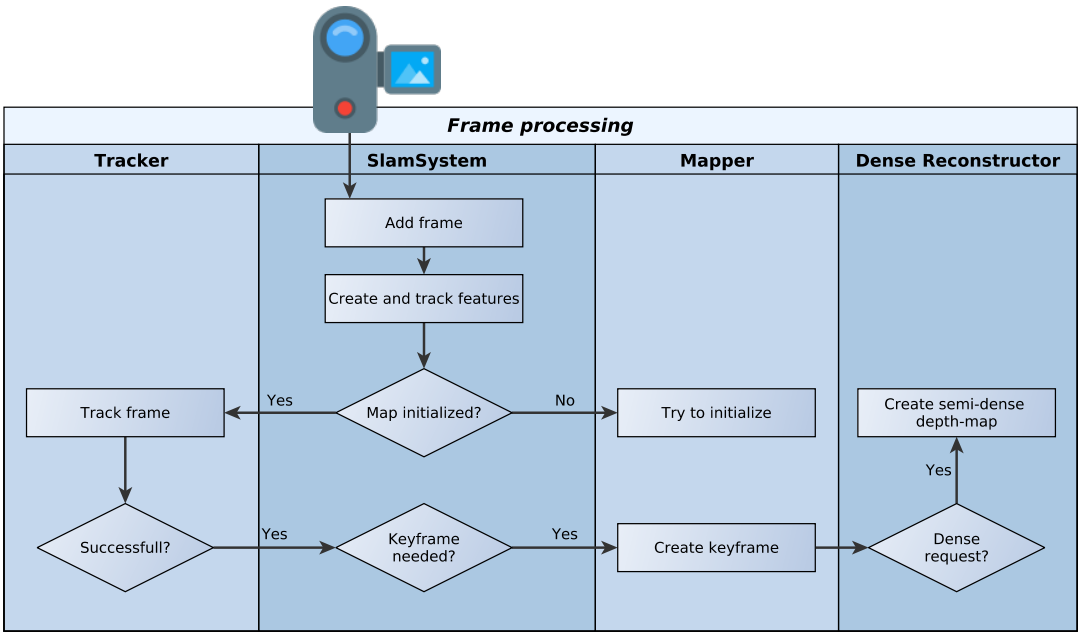


Figure 8.1: Frame processing flowchart. While the *Tracker*, *Slam System* and *Mapper* lane each map to one thread in the application, the *Dense Reconstructor* lane maps to at least one thread. New frames arrive in the system thread and are subsequently processed by the tracking, mapping and semi-dense reconstruction thread, according to the system state.

easy-to-use interface. While the other packages are optional and intended for test purposes, the core package is the only required package to execute CAM-SLAM. Inspired by PTAM and ORB-SLAM, it internally creates separate *tracking* and *mapping threads*. An additional *system thread* is created, which handles communication with the user thread and performs some computations in order to unload the tracking thread. Being able to semi-densely reconstruct the environment, CAM-SLAM starts at least one more *reconstruction thread*. When compiled with OpenMP support, which is optional, this number can be increased arbitrarily. One more *loop-closing thread* is created if enabled[1]. Figure 8.1 illustrates the data flow of a single image frame.

By design, the system thread provides the only interface to the user, and hence, reacts to newly available frames. In contrast to ORB-SLAM, where new frames are grabbed by the tracking thread, image features are computed directly by the

[1]Even though CAM-SLAM is designed to support loop-closing and offers an implementation, it is not yet robust enough to be presented here. For this reason, subsequent experiments are performed without loop-closing.

system thread, outside of the tracking component. Additionally, a simple feature tracker is implemented in the system thread, which is used for map initialization and map point creation. In the original design of CAM-SLAM, the only purpose of the system thread was to present an interface using the *observer pattern*. Such an interface should not be implemented in one of the other time-critical components, because listener notifications are immediately executed in the current thread. Yet, experiments have shown that the system thread is mostly idle when handling events only. As a consequence, further operations were transferred to the system thread with the acceptable drawback that the user should not perform heavy operations in callbacks.

After salient image features were created and assuming that the map is already initialized, the tracking thread takes over the frame. Tracking involves bundle-adjustment of the locally visible map, which is accessible by maintaining a covisibility graph. In the case that tracking was successful, the mapping thread may further process the frame by converting it to a keyframe, if required by the system. Here, the decision of the system is based on the conditions proposed by Mur-Artal et al. [MAMT15] that are discussed in Section 5.3. During mapping, new map points are generated and bad ones are removed. Optionally, the system performs a semi-dense reconstruction of the environment, either online or offline – the latter to increase accuracy.

Also, the above-mentioned data-flow appears to be sequential, all threads are able to execute computations simultaneously. More precisely, while the system is calculating features of a very new frame, the tracking and mapping threads could still be engaged in bundle-adjustment, each on a different (key-)frame.

Listing 8.1 exemplifies the interface of CAM-SLAM. A complete list of all interface methods, as well as a class-diagram of CAM-SLAM, is available in the appendix as Listing B.1 and Figure A.5, respectively.

```
SlamSystem slamSystem;
slamSystem.setup(std::make_shared<ORBHandler>());
slamSystem.setDefaultCameraModel(
    std::make_shared<EquiAngularCamera>(1920, 320)
);
slamSystem.addNewKeyframeListener([](KeyframePointerConst f)
    { std::cout << "Got keyframe: " << f->id; }
);
slamSystem.addNewFrameListener([](FramePointerConst f)
    { std::cout << "Got tracked frame!"; }
);
slamSystem.start();
while(hasFrame()) slamSystem.createFrame(getImage());
```

Listing 8.1: Illustrative usage of CAM-SLAM. After the SlamSystem object is instantiated, a feature handler has to be selected. Using polymorphism, arbitrary feature handlers can be defined outside the core package. Similarly, the default camera model is selected. In theory, CAM-SLAM supports altering camera models – this aspect is untested though and feature tracking would become problematic. The remaining code subscribes to two events, starts the SLAM system and feeds it with images.

8.1.2 Package: cam-slam-tests

Mandatory dependencies: cam-slam, gtest [Inc15]
Optional dependencies: —

The objective of the cam-slam-tests package is twofold: On the one hand, *unit tests* are provided for stability; on the other hand, it is responsible for handling datasets. Supported datasets have either the *TUM-RGBD* [SEE+12], *KITTI* [GLU12], *Catadioptric RGB-D* [SSG14] or a CAM-SLAM specific format. The motivation behind the latter one derives from the ability to recognize synthetic data. For instance, ground truth poses with synthetic image features – subject to a manually defined degree of noise – can be exported from the computer graphics software Blender [Fou15]. Also supported are raw image and video sequences.

This package creates one more thread for each loaded dataset in order to perform asynchronous buffering.

8.1.3 Package: cam-slam-viewer

Mandatory dependencies: cam-slam, cam-slam-tests, Qt [Com15], Boost
Optional dependencies: qcustomplot [Eic15], graphViz [EGK+01], qgv [Nic14]

An interactive graphical user interface for CAM-SLAM is provided in the cam-

slam-viewer package. In addition to the ability to display map points, image features, frames, and keyframes and the covisibility graph; the Qt-based viewer allows to adjust parameters, load datasets, and monitor workload. Further, it includes helpful test- and debug-visualizations, such as the following:

1. Every map point is clickable in the 3D view. After double-clicking a point, all observing keyframes, along with the according image sections, are displayed.
2. Vice versa, image features of the most recent keyframes are clickable, producing the same effect as above.
3. During semi-dense 3D reconstruction, each depth-hypothesis is exemplified by a 3D-line of length $2\sigma_\rho$ and the error function arising from epipolar line sampling (between two manually selected keyframes) is redirected to gnuplot instantaneously. This becomes clearer in the discussion of Section 8.3.3.

Screenshots of the cam-slam-viewer component running CAM-SLAM are available in Appendix A.6 and A.7.

8.2 Data Structures

During run-time, CAM-SLAM occupies most of its time with data. Be it feature matching or bundle-adjustment, data has to be associated and optimized. The most significant data relates to map points, frames and keyframes, and the corresponding data structures are presented next. A graphical overview of map-related structures is provided by Figure A.4. Data arising from implementation concepts, such as mutexes, are omitted.

8.2.1 Map and Map Point

Map points store *position*, *normal* and *color* vectors $\in \mathbb{R}^3$, where the normal is the average viewing direction of observing keyframes. While the map contains a potentially large number of N points, a map point refers to K observing keyframes. Map points additionally exhibit a reliability count that realizes the following idea of ORB-SLAM: A map point must be observed in at least 25% of the frames, where it was predicted to be visible – otherwise it gets deleted. The reliability count of CAM-SLAM behaves more locally, however: After being initialized with the lower bound value $\Lambda_l = 0$, the count is increased or decreased, according to the observation result, but may not exceed the upper bound $\Lambda_u = 8$. Should the variable count fall below Λ_l, the map point is removed. This way, map points associated to non-static objects get removed quickly.

8.2.2 Frame

After receiving a new camera image, CAM-SLAM first assigns a frame object and a camera model to the image. The frame contains the similarity transformation $\boldsymbol{\xi}$ that can – ad libitum – be interpreted as camera pose or camera-to-world transformation. To allow matching, keypoints and descriptors are attached to the frame, in which the keypoints are divided in a grid structure to speed up matching. At this point, the camera model of the frame has to be considered, as discussed for the cylindrical camera model in Section 8.4. As soon as matches are available, they are also associated with the frame.

8.2.3 Keyframe

Each keyframe stores the reference to the frame it is based on. Additionally, and in contrast to other investigated implementations, it references covisible keyframes using a multi-index data structure. Here, the multi-index container combines two red-black trees in order to allow fast sorting based on the covisibility count and fast access based on the referenced keyframe. While the insertion and lookup complexity of this structure is $O(\log n)$, operations on the iterator are performed in constant time. Hence, accessing the N best covisible keyframes has the complexity $O(N)$.

Furthermore, observed map points and an optionally computed inverse depth map are associated with a keyframe.

8.2.4 Camera Model

Since CAM-SLAM supports a wide range of camera models, it requires a uniform interface of shared characteristics. This interface is presented in Listing 8.2. There are only two functions that do not exhibit a default implementation, namely *camToPixel* and *pixelToRay*. While the first one maps from 3D camera coordinates to 2D image coordinates, the latter one performs the reversed mapping – with undefined depth, however. The function *getDistance* computes the offset between two image space coordinates, which can differ from a simple subtraction. For instance, the cylindrical, as well as the equiangular camera model, has to consider the reiterative nature of coordinates ($u_{\min} \boxminus u_{\max} = 1$). *testErrorNormalChi2* checks the plausibility of two image coordinates representing the same point, given the variance of the camera model, using a χ^2 test. The remaining function *eraseArea* allows to mask areas of the image, as illustrated in Figure A.3.

```
virtual Vec2 camToPixel(const Vec3& dir) const = 0;
virtual Vec3 pixelToRay(const Vec2& coord) const = 0;
virtual Vec2 getDistance(const Vec2& coord1,
                         const Vec2& coord2) const;
virtual bool testErrorNormalChi2(const Vec2& coord1,
                                 const Vec2& coord2) const;
virtual void eraseArea(const Mask& mask);
```

Listing 8.2: Common interface for CAM-SLAM camera models. Every central camera model which implements this C++ interface is supported by CAM-SLAM.

8.2.5 Feature Handler

In the same way as with the camera model, the salient image feature detection, extraction and comparison are abstracted using an interface, as shown in Listing 8.3. The interface should be self-explanatory: While *createKeypoints* performs the steps of detection and extraction, *getDistance* measures the matching-quality of two descriptors. CAM-SLAM implements this interface for SURF features and ORB features using OpenCV and also supports the ORB detection and extraction methods provided by ORB-SLAM.

```
virtual void   createKeypoints(Frame* frame);
virtual Scalar getDistance(const cv::Mat& desc1,
                           const cv::Mat& desc2) const;
```

Listing 8.3: Common interface for CAM-SLAM feature handling. When feature detection and extraction algorithms are provided, the implementation of this interface usually only takes a few lines of code.

8.3 Algorithms

CAM-SLAM is strongly inspired by ORB-SLAM and in consequence shares commonalities with the works of Klein and Murray [KM09] as well as Strasdat et al. [SDMK11]. The novelty of CAM-SLAM is that various camera models are supported and for this reason the applicability of algorithms had to be considered carefully during development. As the overall scheme of keypoint-based monocular SLAM methods is introduced in Chapter 5, this section focuses on peculiarities of CAM-SLAM.

8.3.1 Mapping

A central role of the mapping thread is to generate new map points. Section 4.2 discussed available triangulation methods, whereat some of them, like the pop-

ular *linear triangulation*, are restricted to the pinhole camera model. Methods that allow arbitrary camera models include angular triangulation, midpoint triangulation and direct triangulation. During experiments, direct triangulation was rejected rather quickly, as it is more expensive than midpoint triangulation and did not perform better. Tests revealed that CAM-SLAM is able to execute stably with midpoint as well as angular triangulation. This result might be unexpected in the light of Section 4.2, but it becomes clear when considering that the mapping thread performs local bundle-adjustment whenever a new keyframe is created – including the moment of map initialization. As long as a triangulated map point is not rejected as an outlier, which happens more frequently with midpoint triangulation, its position will be optimized by the mapper. With respect to efficiency, midpoint triangulation executes around 60× faster than angular triangulation. Roughly 3000 calls to the triangulation method are performed each second, depending on the type of image features, the camera and the dataset. During experiments, this resulted in an average computation time of $82.46\frac{\text{ms}}{\text{s}}$ for angular and $1.36\frac{\text{ms}}{\text{s}}$ for midpoint triangulation[2].

In order to gain robust triangulation, the following pre- and post-triangulation (distinguished by ◀, ▶) checks are performed for each potential map point:

- ◀ The angle between the back-projection of both image coordinates has to be larger than a threshold λ_α. This way, map points with a high depth uncertainty are effectively rejected. An alternative approach is to parametrize map points by their inverse depth, as discussed in [CDM08].

- ▶ Inspired by ORB-SLAM, the scale consistency is confirmed by verifying that the ratio of distances to both camera centers corresponds to the feature pyramid levels. For instance, when the features are on the same level, the ratio has to be in a range close to 1.

- ▶ The projection of the newly created map point to both image spaces $\boldsymbol{u}_{1,2}$ has to be valid.

- ▶ The error between the keypoint positions and $\boldsymbol{u}_{1,2}$ has to be lower than a threshold λ_e which is based on the image-space covariance of the camera model.

[2]The attentive reader might be surprised to face the "unit" $\frac{\text{ms}}{\text{s}}$. It is equivalent to state that a thread is occupied for 82.46‰ or 1.36‰ of its time. In my opinion, however, the explicit notation used above is more comprehensible, and also gives a better sense for the cost than an absolute measurement which is in the order of microseconds: 28.62 µs vs 0.46 µs here.

Notice that in contrast to regular visual SLAM methods, a maximal re-projection error is enforced instead of the epipolar constraint[3]. Enforcing the epipolar constraint has been avoided, as it would be dependent on a specific camera model and possibly expensive to evaluate.

When CAM-SLAM is newly started or when it lost tracking, it tries to initialize mapping. In the course of this, an essential matrix estimation is performed in a RANSAC-fashion. Even if this approach is less usual than the fundamental matrix estimation, it again offers the advantage of being camera model independent – as long as it is central. The first step of map initialization is to select a reference keyframe, usually based on the first input frame. Afterward, each new frame is treated as a potential keyframe and keypoints are matched to the reference keyframe, which is necessary to employ the 8-point algorithm. The matching is based on a feature tracker that locally searches the best matching descriptor in two immediately consecutive frames for each previously matched keypoint of the older frame. This process is accelerated using the grid structure associated with the camera model, and when a match succeeds it is back-propagated to the reference keyframe. This can be understood as a survival of the fittest scheme, as the number of keypoints available for back-tracing is monotonically decreasing. When the number of matches falls below a threshold, a new reference keyframe is selected.

Due to the local search, camera movements should not be too aggressive until being initialized. It is common practice to pre-define a threshold for good features in order to remove outliers. Since this reduces flexibility – thresholds are only valid for one descriptor type and could also be scene dependent – CAM-SLAM follows another strategy. It is assumed that keypoints that were successfully traced back several frames tend to correspond to good matches and that after $\Lambda_i = 4$ consecutive frames without switching the reference keyframe, descriptor distances are approximately normally distributed around a good threshold. Then, after observing Λ_i consecutive frames, the threshold λ_f is learned based on a generous χ^2 test. As soon as the map is initialized, another χ^2 test is performed to refine the threshold. This scheme has been tested successfully with binary ORB and real-valued SURF descriptors.

When the map is successfully initialized, the mapper performs local bundle-adjustment to optimize map points and keyframe positions. The implementation is based on the g^{2}o graph optimization framework and adopts the double window approach presented in Section 5.3.1. As with ORB-SLAM, the optimization process is performed twice: At a coarse scale first that is with few Levenberg-Marquardt iterations, and at a finer scale after outliers are removed.

[3]One could argue that this is an implicit epipolar check, since the re-projected position is located on the epipolar line.

Map points are created whenever a new keyframe is created. This time, point correspondences are found using FLANN [ML09], and if applicable, also using the tracking method employed during initialization. Points are always triangulated based on a match between keyframes.

8.3.2 Tracking

The tracking-component is responsible for locating the camera in space and has to operate as fast as possible. Should a new image frame arrive before the last one is processed, the new frame is skipped. Initially, the pose of a new frame is estimated using a linear motion model that also incorporates skipped frames. A more sophisticated guess would be possible, for instance with Kalman filtering, but the linear model showed to be sufficient.

Given the initial estimate, feature matching is performed with the currently selected keyframe by projecting available map points to the new frame. This is possible because the keyframe has keypoints associated to map points and map points, as well as the new frame, are already located in 3D space. The matching is again accelerated using the grid structure, and an epipolar check is not performed. The distance between the projected and the actual keypoint position has to fall below a threshold, however, and the descriptor distance is tested using the threshold learned during map initialization. If the number of successful matches is lower than the threshold λ_m, the new frame is labeled as untrackable. In this case, the system tries to create a new keyframe as fast as possible, because this could assist future tracking. After receiving Λ_t untrackable frames in a row, tracking is set to be lost. During experiments $\Lambda_t = 1$ was specified. Hence, tracking was lost at the first untrackable frame.

In the case that enough matches are found, bundle-adjustment is performed to refine the initial pose estimate. This process is again divided into two steps – first, before outlier removal; and second, with outlier removal. In contrast to the bundle-adjustment of the mapping-component, map point positions are unaltered and fewer points are used.

Finally, as with ORB-SLAM, the tracker requests a new keyframe when less than $\Lambda_v N_{fk}$ map points associated to the keyframe were found in the frame, where N_{fk} is the number of map points in the keyframe and $0 < \Lambda_v < 1$, here $\Lambda_v = 0.9$. This can be understood as a visual change condition.

8.3.3 3D Reconstruction

A comprehensive representation of the environment is crucial for mobile robots, and Chapter 6 already introduced related concepts. CAM-SLAM implements the

(a) Target keyframe K **(b)** Reference keyframe K_1 **(c)** Reference keyframe K_8

Figure 8.2: Application of CAM-SLAM's semi-dense reconstruction to the *rgbd_dataset_freiburg3_long_office_household* dataset. The first image shows the target keyframe, and the pixel subject to inverse depth estimation is highlighted by a green dot. The second and third image show the corresponding epipolar line. Red pixels on this line are rejected as their cost is above a threshold and yellow pixels are rejected due to one of the conditions presented in Section 6.1. A green circle marks the matches' position.

procedure presented by Mur-Artal and Tardos in [MAT15] and again derives generalized algorithms in order to support a wide range of camera models.

The method is outlined as follows: Given a keyframe K that is subject to semi-dense 3D reconstruction and a set of covisible keyframes $K_1..K_n$, an inverse depth value is computed for every applicable pixel of K by performing epipolar line matches with $K_1..K_n$. Every match results in a depth hypothesis, and hypotheses are fused if enough are available. As all inverse depth computations are independent, this process is executed in parallel with an arbitrary amount of threads using OpenMP.

Because CAM-SLAM assumes a central camera model, epipolar line searches are possible, but the parametrization of these lines is not predefined. Capturing hyperbolic or parabolic mirrors yields conical epipolar lines, while the cylindrical model results in sinusoid and the pinhole model in straight lines.

CAM-SLAM offers complete independence of the specific type of epipolar lines by employing a line simplification algorithm, and follows the same principle as the Ramer–Douglas–Peucker [Ram72] algorithm. Given the target keyframe K with the inverse depth range $d_{\min} < d < d_{\max}$, a reference keyframe K_i and a pixel coordinate $\boldsymbol{u}$ of interest, the back-projection of $\boldsymbol{u}$ at depth $d_{\min}$ and $d_{\max}$ is projected to the image space of K_i and referred to as $\boldsymbol{v}_{1,2}$. Here, the extremes $d_{\min}$ and $d_{\max}$ are chosen as the 5%- and 95%-quantiles of the depth values of all map points associated with K, which effectively narrows the search and removes outliers. The direct line between $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ is then sub-sampled by projecting

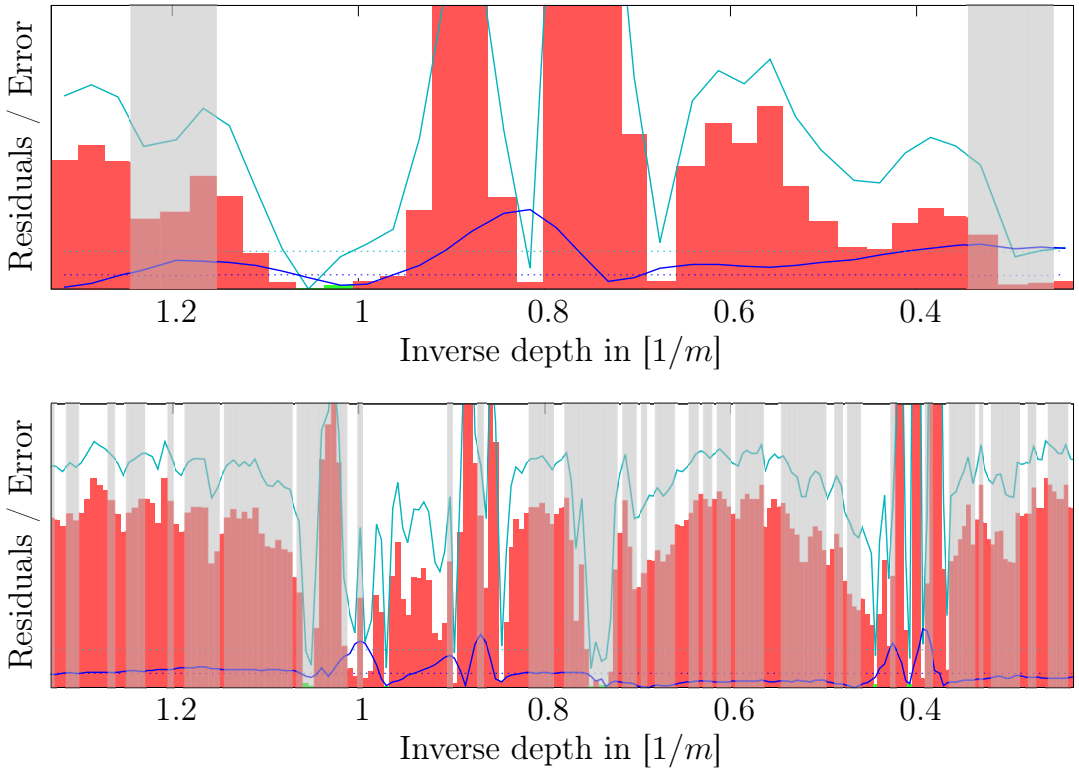
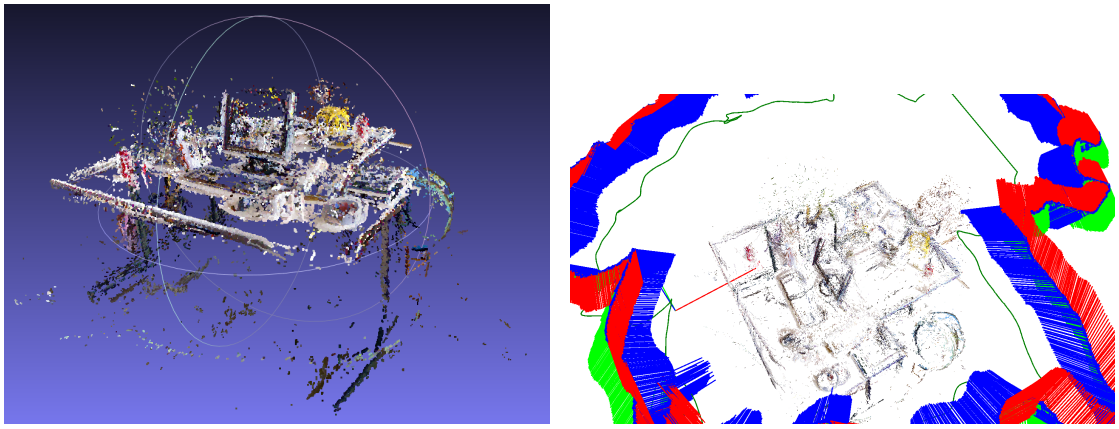


Figure 8.3: Error progression during epipolar line sampling. While the red bars in the upper plot resemble the error of the reference keyframe shown in Figure 8.2b, the bars in the bottom image resemble the one of Figure 8.2c. Further, blue lines account for intensity residuals and turquoise ones for gradient residuals. Grayly marked areas are rejected due to the gradient angle condition (condition 2 of Section 6.1). A version of the upper plot using direct depth instead of inverse depth parametrization is in the appendix as Figure A.1 for the interested reader. The plots demonstrate a well-known property of inverse depth sampling: In the short-baseline case (upper figure) fewer minima exist and the cost-function behaves smoother. For this reason, it is easier to find the optimum, but since pixels correspond to a larger inverse depth range, the result is less precise. For larger baselines, the precision will increase but the task of finding the correct match becomes more serious.



(a) Visualization in Meshlab **(b)** Visualization in the cam-slam-viewer

Figure 8.4: Result of the semi-dense triangulation procedure of CAM-SLAM. At a closer inspection it becomes apparent that some edges are reconstructed twice. Please note, however, that this experiment has been performed without loop-closing. As a result, each inaccuracy that emerged while executing monocular SLAM directly affects the quality of the 3D reconstruction. Hence, the monocular SLAM procedure, as well as the semi-dense 3D reconstruction procedure, can be considered to be of high quality.

intermediate 3D points of the declared depth range to K_i. Sub-sampling is repeated iteratively until subsequent line-segments are nearly straight or until the length of segments falls below a threshold. The pseudo-code for this algorithm is shown in Listing B.2. Please notice that this procedure involves little overhead when used in conjunction with pinhole cameras, since the sub-sampling stops at the first iteration.

CAM-SLAM samples the epipolar line segments using the Bresenham [Bre65] algorithm and assigns a matching cost to each pixel, as in Equation 6.3. It also performs the sub-pixel refinement and error propagation discussed in Section 6.1. Figure 8.2 illustrates the epipolar line sampling for one target and two reference keyframes. The error and residual values belonging to the sampling process in these images are plotted in Figure 8.3. While conditions 1 and 2 of Section 6.1 are enforced during sampling, condition 3 is not, as it makes assumptions about the camera model.

After completing the semi-dense inverse depth-map, hole filling and regularization are applied to improve the quality. These steps are realized comparably to the LSD-SLAM implementation and are repeated $\Lambda_{N_R} = 2$ times. The hole-filling step iterates over the inverse depth map and analyzes a window of size $\Lambda_h = 3$ around each pixel without depth value. When more than $\Lambda_{h_n} = 3$ neighbors in the window have an inverse depth value and have a similar intensity to the pixel of interest, their merged value is assigned to the pixel, as in Equation 6.8. In contrast

Figure 8.5: The image on the left side shows the V.360° camera. Courtesy of VSN Mobil, retrieved from [Mob15]. On the right hand side 40% of a picture taken with the V.360° is depicted. The cropping is due to the extreme (6480×1080) aspect ratio of the image. The full picture is available in A.10.

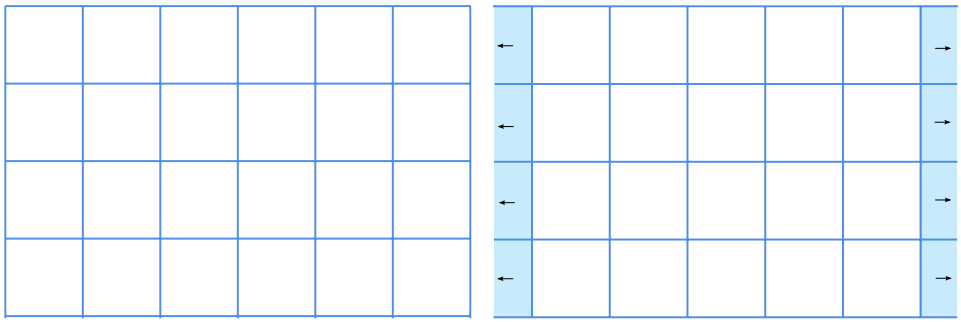
to the hole-filling step, the regularization step considers a window (of size $\Lambda_r = 5$) around each pixel that has a value assigned. LSD-SLAM then averages those values in the window that are not further away than $2\sigma_f$ from the current one. CAM-SLAM, on the other hand, performs a χ^2 test at 95%, which is effectively the same. The reason to prefer the χ^2 here is that inverse variances of inverse depth values are stored in the implementation. As a result, the χ^2 – in contrast to a direct σ_f test – can be performed without performing computational expensive divisions. CAM-SLAM again incorporates image intensities and merges the inverse depth values as during hole filling.

Inverse depth map regularization and hole filling are speeded up using the multi-threading capabilities of OpenMP. A result of the triangulation procedure is shown in Figure 8.4.

8.4 VSN V.360°

Among other cameras, the V.360° of VSN Mobil was used during experiments. It is a consumer class camera and is displayed in Figure 8.5. The initial idea was to equip a mobile robot with this camera in order to gain omnidirectional vision[4]. For this reason, the camera had to provide a static video stream via HDMI – a functionality that was not provided by the up-to-date firmware at the beginning of this thesis. To overcome this limitation, a new remote control software with Python and C++ interface has been developed, using reverse engineering techniques. The course of action is only briefly outlined here: The official application

[4]The camera supplies other sensor data that is useful for robots but of no interest here.



(a) Regular 6x4 grid **(b)** Reiterative 6x4 grid

Figure 8.6: Comparison of a regular grid and reiterative grid. While the extreme columns have maximal distance (here $\|x_{\min} - x_{\max}\|=5$) in the regular grid, their distance is 1 in the reiterative grid, where the maximal horizontal distance amounts to $\frac{w}{2} = 3$ and horizontal distances in general are given by $\triangle\, x = \min(\|x_2 - x_1\|, w - \|x_2 - x_1\|)$. In the implementation of the reiterative grid, 50% of the first column belongs to the beginning and the remaining 50% to the end of the underlying image.

communicates with the camera via wifi[5], and therefore, a man-in-the-middle attack [CH15] suggests itself. The major hurdle is that the manufacturer makes use of certificate-pinning to secure the communication. This problem has been overcome by decompiling the official application and injecting a fake certificate. Large portions of the extracted protocol are included in the remote control software, which is available on github [Rü15].

As already mentioned in Section 3.2.5, the V.360° exhibits a cylindrical camera model. Because tracking, as well as mapping, requires a grid structure for feature matching, the lateral cylinder surface must be divided in a reiterative grid to preserve locality, as illustrated in Figure 8.6.

8.4.1 Calibration

The main purpose of camera calibration is to find the intrinsic parameters of the applied camera model. Due to the popularity of traditional cameras, this process is well studied for the pinhole camera model. The fact that various omnidirectional camera models are available – Section 3.2.3 presents an established one – is reflected in the calibration methods: They are specialized for the model at hand. As with traditional cameras, automatic omnidirectional calibration methods were

[5]The camera also communicates via Bluetooth and turning the device on is only possible with Bluetooth. The new open-source remote control supports turning the device on and then handles all communication via wifi.

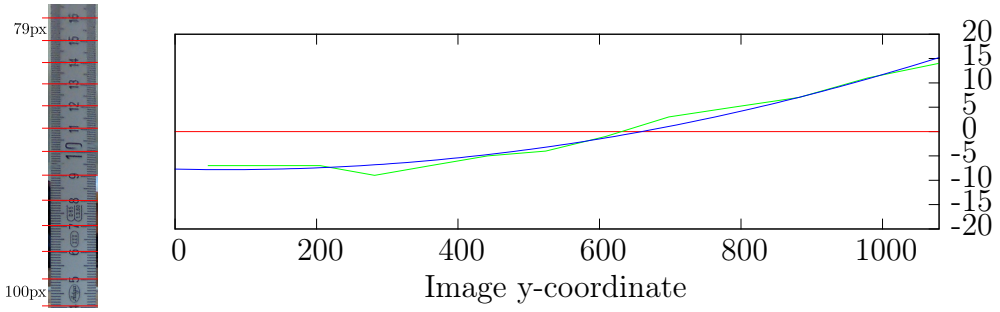


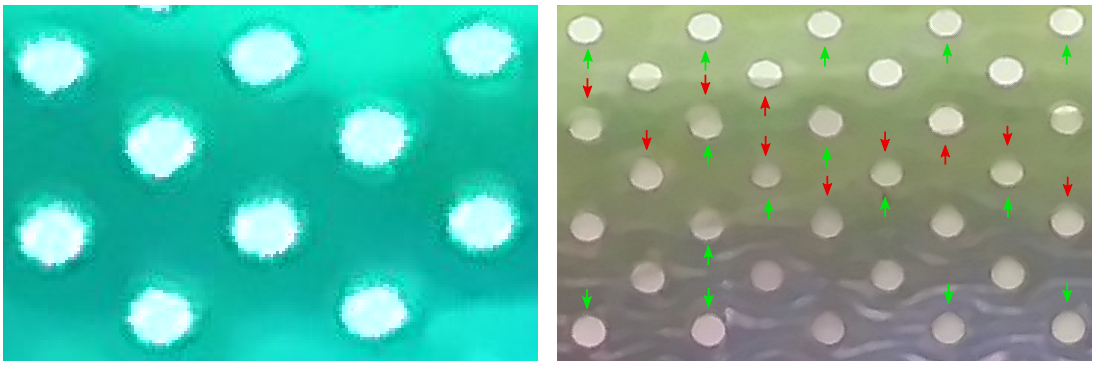
Figure 8.7: The ruler on the left side was aligned with the camera axis in order to calculate the pixel per cm ratio at several stops. The average ratio resulted in $86\frac{px}{cm}$, while the ratio at the top yielded $79\frac{px}{cm}$ and at the bottom $100\frac{px}{cm}$. This behavior has been measured using distinct angles. It occurred in every direction, with varying intensity though. A reason for this peculiarity could be that the manufacturer compensates for the decreasing resolution towards the mirror's center this way. The green plot on the right represents the measured ratio-divergence from the average. It has been approximated by an second order function, which is shown in blue and used to diminish this effect. The red line at 0 would correspond to optimal measurements of a perfect cylindrical camera.

proposed, such as by Bazin et al. [BKDV08]. Adan Salazar-Garibay compiled available methods for a survey in his PhD thesis [SG11]. Unfortunately, it is not possible to access raw images of the V.360° and in consequence, available calibration software could not be used.

In order to acquire calibration parameters nonetheless, a two-stage approach was chosen. As usual, grid points of calibration patterns were extracted and used to formulate a cost-function based on extrinsic and intrinsic camera parameters by comparing actual and predicted grid-point positions. Since the cylindrical camera model is highly non-linear, the first step has been to identify a set of promising local minima by an evolutionary computation system. This decision had practical reasons as well: An adequate system had been developed previously and the formulation of an appropriate fitness function is trivial. The second step has been to refine the results numerically using a non-linear optimization algorithm, namely the quasi-Newton BFGS [NN91] method[6].

An additional calibration approach has been tested that could not improve the calibration results, however: The camera was placed inside a metal tube with a well-structured pattern of round holes in it. The center of each hole was detected and compared to a projected position, as usual. The advantage of this method is that only one image is required, and hence, that extrinsic parameters have to be

[6]This is definitely not the fastest method, nor the most elegant one, to perform camera calibration. It rather resembles a general non-linear optimization approach.



(a) 180 × 120 image detail **(b)** 555 × 380 image detail

Figure 8.8: Illustration of image artifacts produced by the V.360°. At difficult lighting conditions, the camera tends to produce scratchy edges as in 8.8a. Additionally, edges might suffer blurriness, as the redly marked ones in 8.8b, even though vicinal ones appear sharp, as the greenly marked ones. Both effects are most probably related to post-processing steps performed directly on the camera.

estimated only once. This is possible as the pattern of holes is spread over the complete image space. Figure A.9 shows a camera image from within the tube and with detected circles.

Please notice that camera calibration is a domain of its own and an in-depth discourse is out of the scope of this thesis.

8.4.2 Artifacts

That VSN aims at the consumer market with the V.360° already became apparent during calibration, which an image quality analysis reconfirms. Figure 8.8 highlights two distinct artifacts that are frequently observed, especially at difficult lighting conditions.

It is likely that these artifacts are related to post-processing steps, as these are commonly performed on consumer devices such as mobile phones. These effects appear with a recently cleaned camera, and an option to turn post-processing off is not provided. Further, and again depending on the lighting conditions, the video stream suffers from considerable motion blur and an unpreventable auto-adjustment of the frame-rate is performed. During experiments, the artifacts were addressed by down-sampling the footage. Down-sampling removes the high-frequency portion of the signal, and hence, reduces some of the effects discussed above. Here, the insertion of new aliasing-artifacts has to be prevented, for example by applying Lanczos interpolation or bilinear interpolation.

Chapter 9

Results

In order to evaluate the performance of CAM-SLAM in terms of accuracy, speed and robustness, a variety of tests has been carried out. The usual approach for such tests is to employ datasets with highly accurate ground-truth data and to compare it to generated data. As flexibility is the major objective of CAM-SLAM, experiments have been executed on diverse datasets and omnidirectional data has explicitly been included. The used datasets are characterized as follows:

1. TUM-RGBD datasets[SEE+12]: Pinhole camera, handheld
2. KITTI datasets[GLU12]: Pinhole camera, mounted on moving car
3. Catadioptric RGB-D dataset[SSG14]: Catadioptric camera, mounted on moving car
4. *V360 dataset*: Cylindrical camera, handheld
5. *Room dataset*: Equiangular camera, synthetic images and motion
6. *Mars dataset*: Equiangular camera, synthetic images and motion

While datasets 1-3 are publicly available, sets 4-6 were created within the scope of this thesis. The setup for acquiring set 4 is shown in Figure 9.1. The V.360° was rigidly coupled with a marker and tracked using a slightly modified version of Aruco 1.3[1] [GJMSMCMJ14]. Although this tracking is not extremely accurate, it suffices to perform a qualitative evaluation of running CAM-SLAM with the

[1]Modifications were necessary to increase the tracking performance on a high-resolution camera (GoPro HERO3). While approximately 70% of frames were tracked before the modification, 100% were tracked afterward.

(a) Back view **(b)** Front view **(c)** Tarcked camera

Figure 9.1: Setup used to track the V.360°. In order to create sequences with associated ground-truth, the camera motion has been calculated using marker tracking.

Figure 9.2: Rendering of the synthetic room scene. This image is used for illustration purposes only – one frame of the equiangular renderings that have been used during experiments is provided in Appendix A.8.

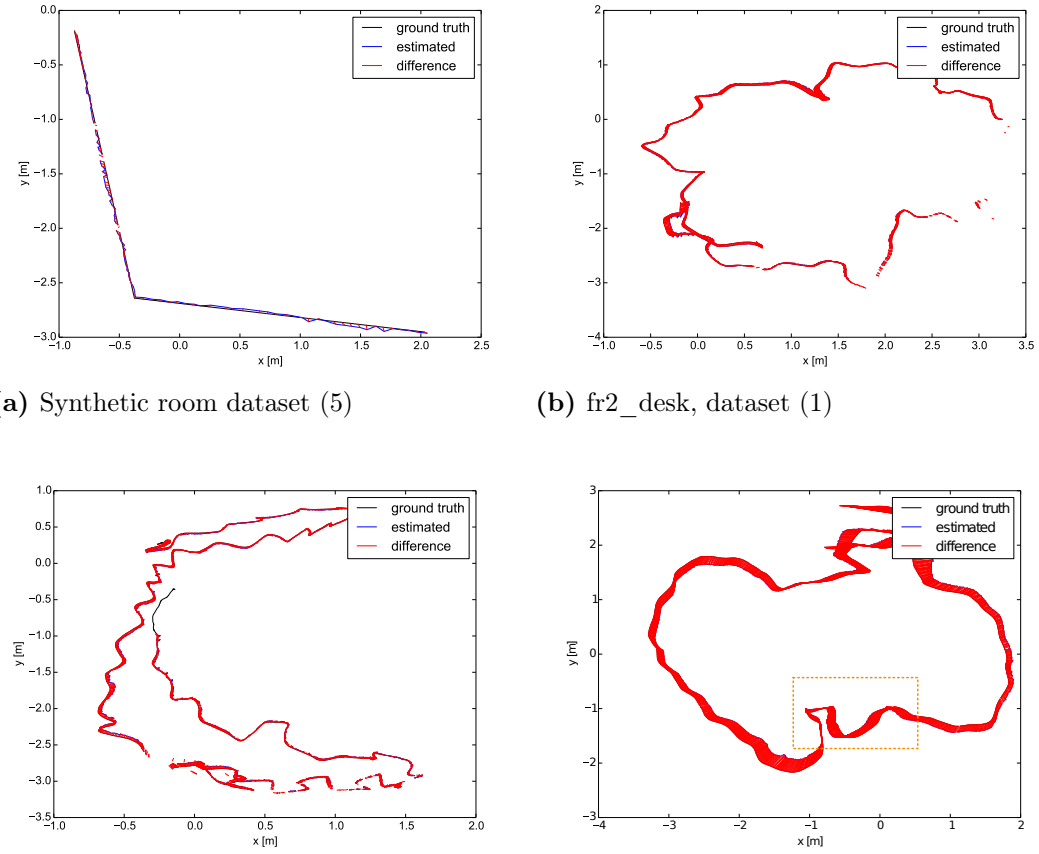
	CAM-SLAM	PTAM	LSD-SLAM	ORB-SLAM
fr1_xyz	2.60	1.15	9.00	**0.90**
fr2_desk	3.53	×	4.57	**0.88**
fr2_desk_person	2.29	×	31.73	**0.63**
fr3_long_office	16.83	×	38.53	**3.45**
synth_room	**2.84**	×	×	×
v360 sequence1	**16.21**	×	×	×
v360 sequence2	**14.92**	×	×	×

Table 9.1: Comparison of root mean square absolute trajectory errors in *cm*, using small-scale datasets. With the exception of CAM-SLAM measurements, the data is provided by Mur-Artal et al. [MAMT15].

V.360°, as discussed later. Datasets 5 and 6 were generated using the computer graphics software Blender with an equiangular camera model. Synthetic datasets have the advantage of providing perfect ground-truth trajectories, but introduce appearance-related difficulties. For instance, while most real surfaces exhibit structure due to imperfections, synthetic ones can be completely smooth, and hence, featureless.

9.1 Accuracy

Experiments are grouped in two categories: Small- and large-scale experiments. A good measure of performance for the former category is the *absolute trajectory error* (ATE), as described by Sturm et al. [SEE+12]. It is computed by determining the absolute differences of camera positions, after aligning the ground-truth with the SLAM-produced trajectory. Here, either a 6D- or 7D-alignment is performed, for example, using the method of Horn [HHN88]. The absolute trajectory error is less suited for large-scale data, however. Imagine a SLAM system always produced the perfect trajectory, but failed in one curve so that every subsequent position had a major offset to the ground truth. Compare this to the case in which a system continuously produces errors, but luckily, did not misinterpret the rotation. In a large-scale scenario, the second system might obtain an ATE which is 1000× less than the system that made one mistake only. Hence, a comparison of *relative pose errors* (RPE) that effectively measure the local drift is more meaningful on large-scale data. As CAM-SLAM – just like other monocular SLAM-methods – suffers from scale-drift, only segments of large-scale datasets are investigated. For a more profound evaluation on large-scale data, loop-closing would be required, which has not been performed during experiments.



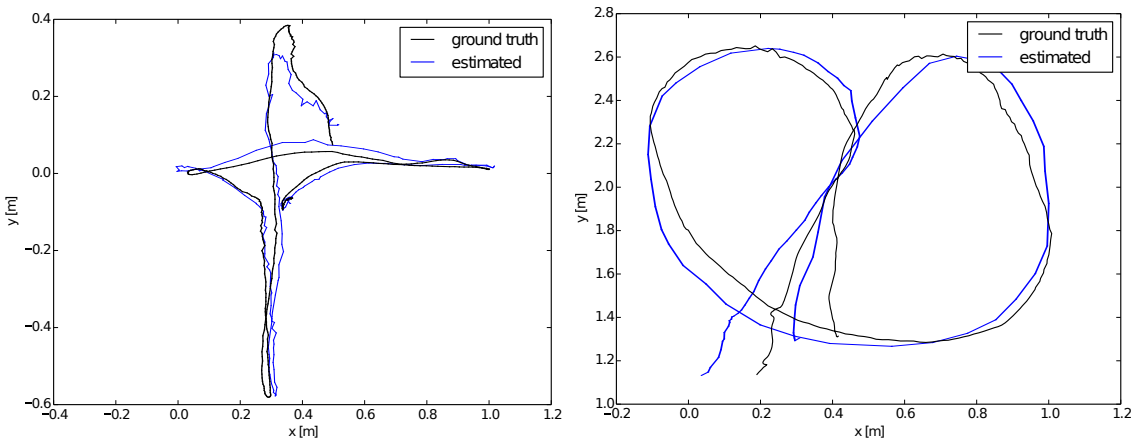
(a) Synthetic room dataset (5)

(b) fr2_desk, dataset (1)

(c) fr2_desk_person, dataset (1)

(d) fr3_long_office, dataset (1)

Figure 9.3: Plots that highlight the differences between CAM-SLAM estimated trajectories and ground-truth trajectories. While ground-truth trajectories are black, estimated trajectories are blue and differences between associated positions are plotted red. Gaps in the graphs correspond to missing associations, mainly because ground-truth data is missing.



(a) V.360° sequence1 (6) **(b)** V.360° sequence2 (6)

Figure 9.4: Plots of two V.360° sequences, where the ground truth trajectories are black and the estimated trajectories are blue. While SURF-features were used in 9.4a, ORB-features were used in 9.4b. That the ground-truth trajectories are not highly accurate becomes apparent when observing their discontinuities.

Table 9.1 compares the *root mean squared* ATE produced by CAM-SLAM to the ones produced by PTAM, LSD-SLAM and ORB-SLAM. Sequences starting with *fr...* belong to the popular TUM-RGBD benchmarking datasets. While the sequences *fr1_xyz* and *fr2_desk* present a static environment with a moving camera, a person is interacting in sequence *fr2_desk_person*. Processing non-static environments is difficult for visual SLAM systems, which is shown in PTAM failing to handle the scene and LSD-SLAM producing large errors. In contrast, the reliability count presented in Section 8.2.1 and the outlier removal of ORB-SLAM proved to be effective. Figures 9.3b and 9.3c show related plottings of the ground-truth and estimated trajectories as well as their differences. The scene *fr3_long_office*, which also presents a moving camera in a static environment, is difficult because the camera maneuvers close to an object (the bear), rotates and maneuvers back. This can easily cause scale-drifts or even tear down tracking and is liable for CAM-SLAM's high error in this scene. The difficult part of sequence *fr3_long_office* is outlined with an orange box in Figure 9.3d.

The results presented in Table 9.1 show that CAM-SLAM has a comparable performance as state-of-the-art methods. It outperformed LSD-SLAM in all test-sequences and it was robuster than PTAM. ORB-SLAM, on the other hand, consistently produced better results than CAM-SLAM, which has several causes:

1. ORB-SLAM is more finely tuned with respect to the camera model and keypoint descriptors.

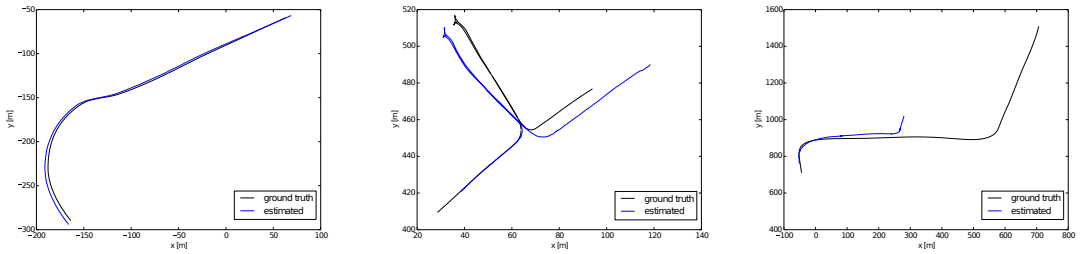
2. ORB-SLAM performs loop-closing, while CAM-SLAM did not in the experiments.

3. ORB-SLAM tracks a local map. Up to now, CAM-SLAM uses the local map for mapping only and performs tracking to a reference keyframe.

Nevertheless, CAM-SLAM benefits from its flexibility. Neither of the other methods is able to handle omnidirectional data, while CAM-SLAM performed as well on the omnidirectional synthetic room sequence as on the other sequences. The estimated trajectory is shown in Figure 9.3a: The camera firstly moved straight from one corner of the room to another one, started rotating and moved to yet another corner while rotating. CAM-SLAM worked well on this set and produced errors of the same order as with the TUM pinhole-datasets.

Figure 9.4 presents the results generated with the V.360°-sequences. Even though the ground-truth is not highly accurate, it is obvious that CAM-SLAM was able to reproduce the original trajectory. Absolute trajectory errors are included in Table 9.1. Here, the values are similar to the one of sequence *fr3_long_office*. There are three reasons for the relatively large error: First and foremost, the camera calibration could not be performed perfectly, as the V.360°-model is not completely in accordance with a cylindrical model. This effect is illustrated in Figure 8.7 and it has been reduced using a second order approximation. Additionally to the inaccuracies of the camera model, the artifacts discussed in Section 8.4.2 further encourage trajectory errors. The last reason for the relatively high error is that the ground-truth itself is prone to imprecision. Considering these circumstances, CAM-SLAM's performance on the V.360°-dataset can be regarded as being convincing.

In addition to evaluating the accuracy of CAM-SLAM in small-scale sequences, the applicability to large-scale sequences has been tested. Figure 9.5 presents the corresponding trajectories, which were generated using three different camera models. A rendering of the synthetic Mars scene, which was used for the first sequence, is available in Appendix A.2. The according ground-truth trajectory is evenly approximated by CAM-SLAM, despite a modest drift. Due to the more complex camera movement, a more serious drift occurred in the catadioptric dataset. That the estimated trajectory suffers from a drift in scale becomes clear when observing the pulling in and out of the dead-end road in Figure 9.5b. One has to consider, however, that the catadioptric camera is not mounted centrally on the car and that a certain offset is expected for this reason. Appendix A.3 shows two frames of this dataset from two different cameras. The output of only one of these cameras was used during experiments. The remaining plot shown in Figure 9.5c resembles experiments with the KITTI dataset that uses a pinhole camera. In this dataset,



(a) Synthetic Mars dataset (6) **(b)** Catadioptric dataset (3) **(c)** KITTI dataset (2)

Figure 9.5: Plotting of large-scale trajectories.

	ORB			SURF			
	Extr.	Tracking	Mapping	Extr.	Tracking	Mapping	Dense
Desk	18.54	22.02	448.74	325.03	23.07	333.34	384.48
Room	26.43	21.31	352.47	266.30	29.57	266.97	14340
Cata	28.67	8.07	178.12	225.00	33.22	401.47	-
Mars	36.89	18.66	226.03	210.55	7.90	83.83	-

Table 9.2: Timings of several components of CAM-SLAM, while being executed with different datasets. Each column denotes the recording of one component (feature extraction in combination with feature tracking, frame tracking, local mapping and semi-dense reconstruction) and each row denotes another dataset (two small-scale sets: fr2_desk (1), synthetic room (5) and two large-scale sets: catadioptric set (3) and Mars set (6)). While feature extraction and frame tracking timings measure the time for one frame to be processed, mapping and semi-dense reconstruction measurements apply to keyframes. All measurements are are specified in *milliseconds*.

the most severe drift occurred, which might be related to the shorter visibility of map points. Interestingly, SURF feature matching was more reliable as ORB matching, given catadioptric images. This could be related to the higher degree of distortion, which clearly affects keypoint descriptors, but an in-depth analysis remains outstanding.

9.2 Performance

The performance of CAM-SLAM with respect to execution speed has been analyzed by recording timings of essential components. In addition, poor-man-profiling was performed to identify bottlenecks and to understand the overall performance of the system. The results of this profiling step are visualized as a flaming

graph in Figure 9.6, where they are also discussed. Table 9.2, on the other hand, presents the time recordings. An input video-stream with $24 - 30$ frames per second grants a visual SLAM system approximately $42 - 33$ ms to process the frame, when frame drops ought to be prevented. CAM-SLAM fulfilled this constraint for each dataset, when ORB keypoints were used. At this point, it became useful to separate frame tracking and feature extraction, as outlined in Figure 8.1. Ultimately, this is pipelining and allows to run CAM-SLAM at higher frame rates[2]. It is surprising at first that frame tracking and local mapping are executed faster on omnidirectional datasets than on the pinhole camera sequence. The reason for this is that more keypoints were successfully matched with the pinhole camera using ORB-features. As a consequence, more map points were triangulated and involved optimization operations became more expensive. The tracking and mapping of the Mars-dataset has been as fast with SURF-features for the same reason.

A strong contrast between ORB- and SURF-features is given by the respective extraction time. It takes approximately $10\times$ longer to extract SURF-features, which has implications for their applicability: Sequences with aggressive or non-continuous movements, such as sequences generated with a hand-held camera, are more likely to fail with SURF-features. This is because the tracking thread predicts camera poses less frequently, and hence, produces larger deviations in the predictions.

Another contrast becomes apparent when correlating semi-dense reconstruction timings. This process takes significantly longer with omnidirectional imagery. The increased processing time is caused by the line-simplification algorithm, which is discussed in Section 8.3.3. It produces more depth samples when used with non-linear camera models and thus is most efficient on the pinhole camera model.

While Table 9.2 showed that camera tracking can run in real-time, the local mapping procedure is only executed a few times per second. This is expected and is comparable to the behavior of ORB-SLAM. Mur-Artal et al. [MAMT15] measured that ORB-SLAM requires approximately 31ms for feature extracting and frame tracking, which is performed sequentially and approximately 460ms for local mapping. It is unclear, however, which hardware has been used for those measurements. Considering the time required for ORB-feature extraction (they measured 11.42ms for the same method), it must have been about twice as fast. Here, an Intel® Core™ i7-920 processor from 2008 has been used with 1066MHz DDR3-RAM.

[2]Currently, keypoint extraction is performed in the system thread of CAM-SLAM, which means that the execution of call-backs and feature extraction are bidding for the same resource. Hence, call-backs are delayed when using SURF-features and expensive call-backs interfere with the SLAM-system.

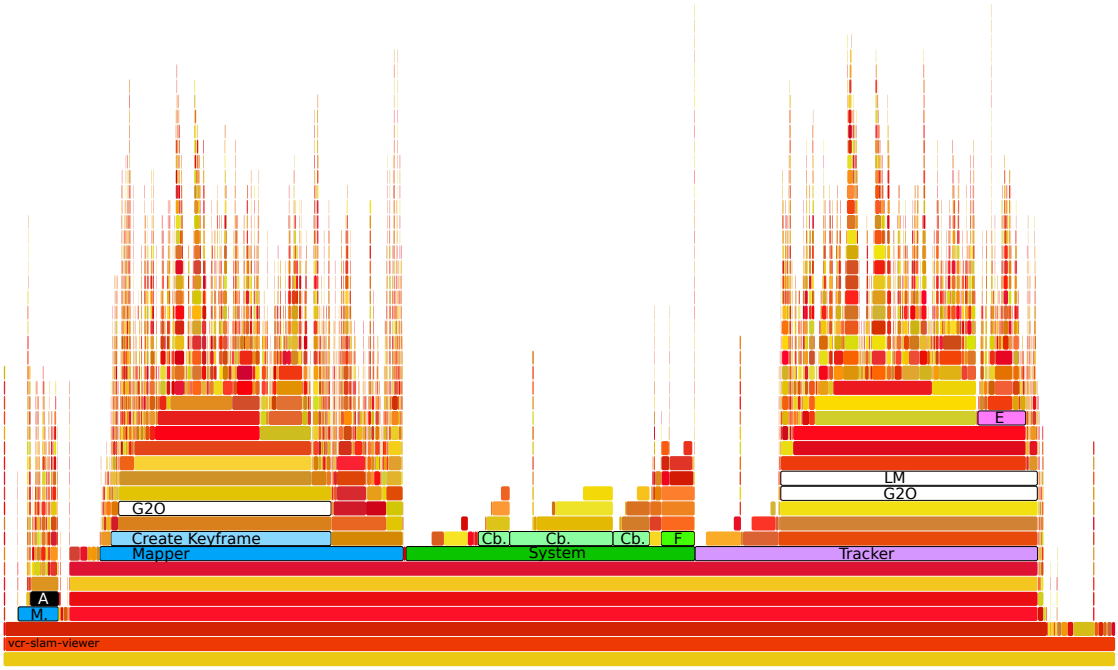


Figure 9.6: Flaming graph of CAM-SLAM. Flaming graphs are visualizations of poor-man-profiling. Basically, the profiler takes snapshots of the program stack at a high rate – here, several thousand times in total – and analyzes the frequency of function calls. As computational expensive functions appear often in the stack, they are likely to correspond to a high frequency. Poor-man-profiler are easy to use, entail little overhead and provide useful stack-information. Here, the main components mapper, system and tracker are highlighted in blue, green and purple respectively. Highlighted in light blue, the mapper spends most of its time with keyframe creation, which encompasses local bundle-adjustment (performed by g^2o and highlighted in white). Due to technical reasons, angular triangulation is shown on a separate stack on the left side, marked in black. Midpoint triangulation would barely appear in the graph. Three of the lighter green calls on the system stack correspond to call-backs, while the slightly more intense greenly marked call represents feature extraction. The tracking thread also spends most of its time executing g^2o which then executes Levenberg-Marquardt optimization. A bit further above on the tracker stack, highlighted in pink, there is the computation of the reprojection error. Here, a pinhole camera has been used. The bar belonging to the reprojection error would be bigger for more complex camera models. Using the information of this graph, one can carefully consider which functions respond best to optimization. Namely, functions that appear frequently, whereas laborious optimizations of functions that are only represented narrowly here should be avoided. In this experiment, the system and tracking threads nearly operate at full capacity, while the mapping thread is fully loaded.

Chapter 10

Conclusion

This thesis presented principles of monocular SLAM techniques, elaborated on state-of-the-art methods and introduced a new system called CAM-SLAM. While CAM-SLAM showed a similar accuracy as state-of-the-art monocular SLAM methods, it provides considerably more flexibility at the same time.

Being research software though, CAM-SLAM can still be improved. First of all, the initialization procedure assumes a non-planar environment. Given a pinhole camera model, it is possible to compute a homography to initialize from a planar environment, but this concept does not generalize to any central camera model. Zhang et al. [ZLZH10] presented an equivalent approach for some catadioptric camera models, but a more flexible method would be desired. Also, CAM-SLAM lacks a re-localization algorithm, and loop-closing has to be implemented more robustly. This would substantially improve the performance on large-scale datasets and simplify the comparison with other methods.

Banissi and Golipour [BG14] recently proposed an algorithm for the efficient sampling of conics. It is possible that this algorithm could reduce the execution time of the semi-dense reconstruction method for catadioptric camera models.

Mur-Artal et al. [MAMT15] already realized that their method might profit from representing points at infinity, as described by Civera et al. [CDM08]. The same argumentation holds true for CAM-SLAM: Especially on sequences that exhibit camera transformations with a rotational component only, an inverse depth representation during tracking can stabilize the SLAM execution.

Another area for future research involves the creation of high-quality omnidirectional datasets. Such datasets are available for traditional cameras and an equivalent benchmarking set for omnidirectional camera models would ease the evaluation process.

To my knowledge, CAM-SLAM is the first system able to support any type of central camera model. In addition to executing monocular SLAM, it is also able to perform a semi-dense reconstruction of the environment. Successful experiments have been performed using synthetic datasets as well as real datasets with a variety of camera models. The sets included small-scale indoor scenes and large-scale outdoor scenes.

In order to obtain CAM-SLAM's flexibility, relevant techniques were analyzed thoroughly. Different triangulation methods were implemented and compared for instance, to decide which ones are best suited for camera-agnostic monocular SLAM. Furthermore, a camera model-agnostic initialization procedure was presented that is also independent of the type of salient images features.

As a result, CAM-SLAM is able to operate with the V.360°, which exhibits a rather uncommon camera model and could not be calibrated with high accuracy. CAM-SLAM is easy to use, avoids heavy weight dependencies and implementing a new camera model can usually be done within minutes.

These characteristics are unique and the potential of CAM-SLAM is considerable. With the help of CAM-SLAM, after being put into a new environment, a machine can quickly recognize its new surroundings, imposing minimal requirements on the camera model.

Appendix A

Additional Figures

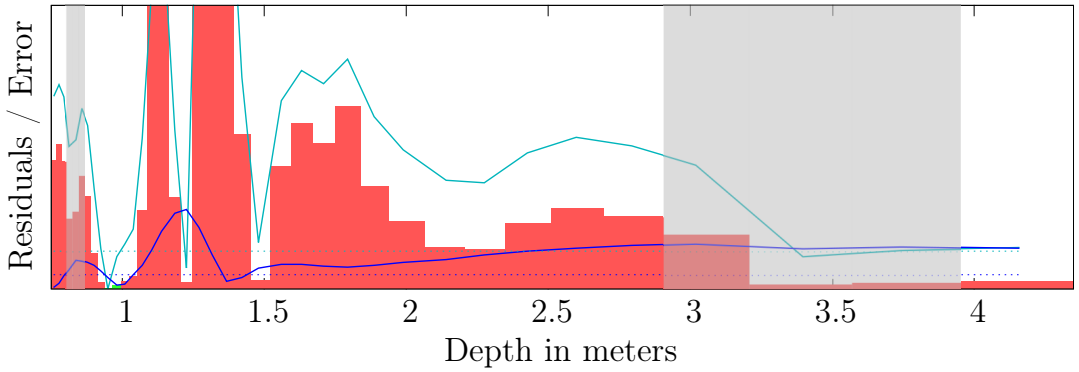


Figure A.1: This figure shows the same epipolar sampling as Figure 8.3, but with a different x-axis scaling. While Figure 8.3 is parametrized by inverse depth, a parametrization by immediate depth values is illustrated here. By comparing the width of error bars, the following relation becomes apparent: The smaller the depth value of a pixel, the smaller is the depth range of the pixel. This effect is neglectable when the inverse depth parametrization is used.

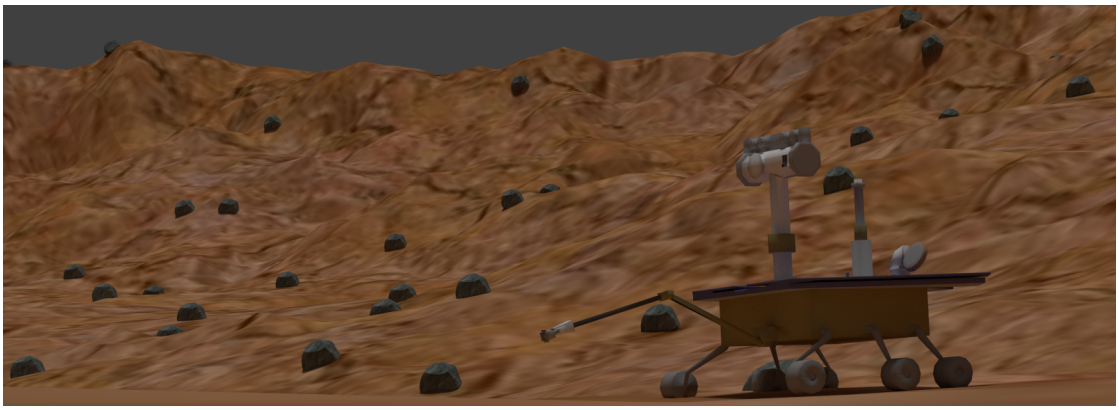
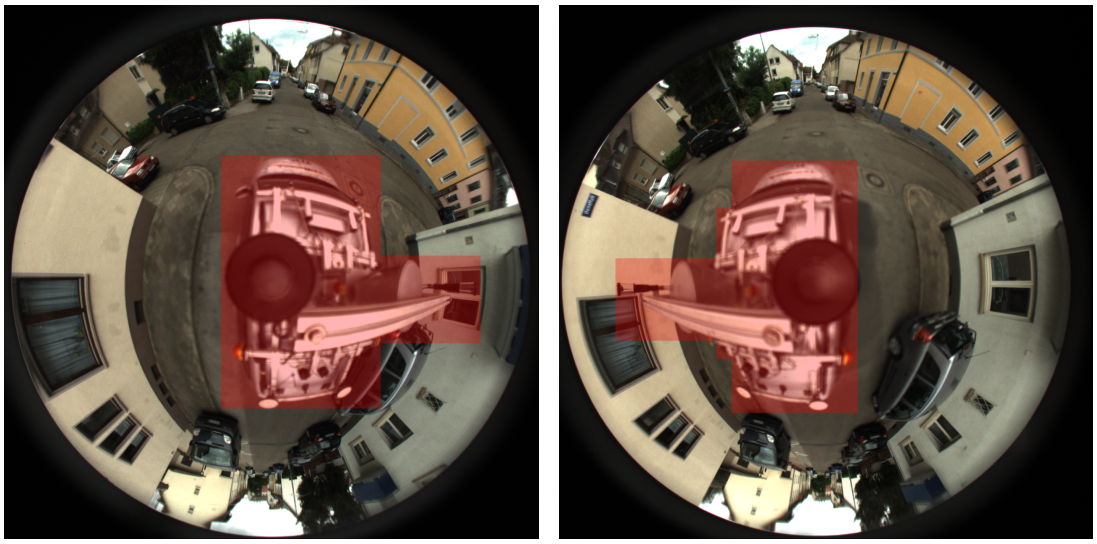


Figure A.2: Rendering of the synthetic Mars scene. As with the rendering shown in Figure 9.2, this rendering is used for illustration purposes only and an actual equiangular rendering would look like Figure A.8.



(a) Left camera **(b)** Right camera

Figure A.3: Two simultaneously acquired frames of the catadioptric dataset [SSG14] (3) that was used during experiments. When omnidirectional cameras are used, it is common that a considerable portion of the images is occluded by the robot – or car in this case. Hence, an appropriate masking of the image frames is mandatory.

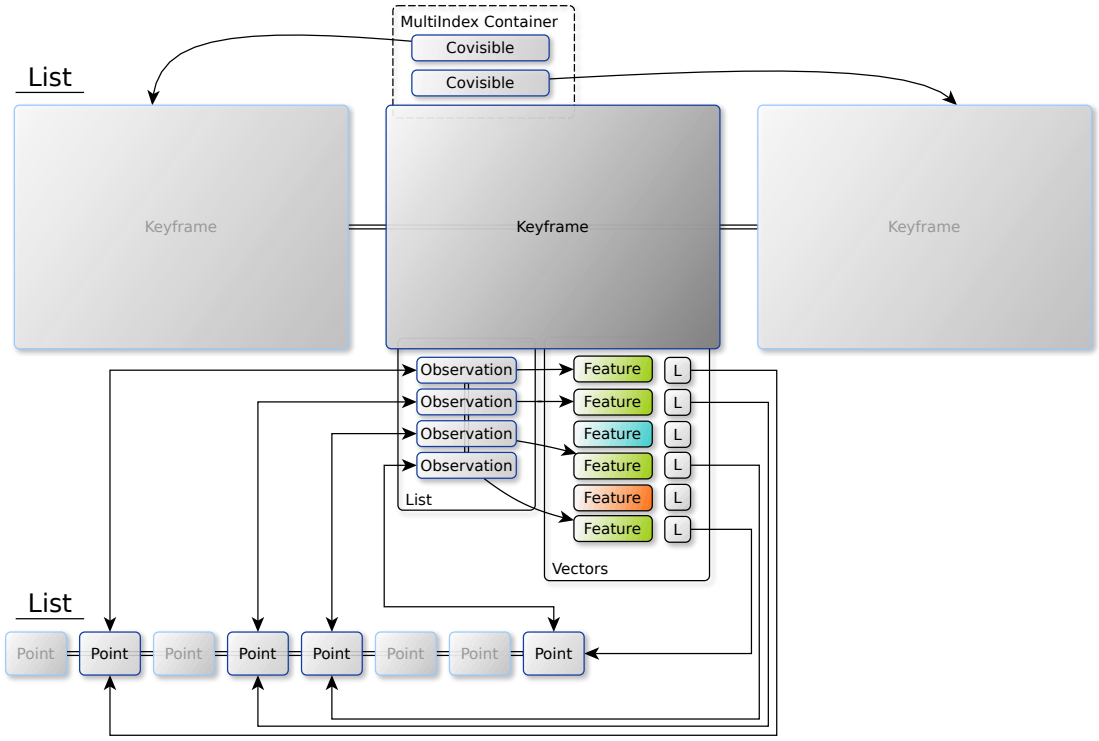


Figure A.4: Map data-structure of CAM-SLAM. The map contains a list of keyframes and map points, which offers the ability to add or remove instances in constant time, or $O(1)$. Each keyframe also stores a lookup-table of the size of image features, in order to quickly associate features to map points. Map points, on the other hand, reference their observations, which in turn allows for quickly associating keyframes to map points. The list of observations that is maintained by every keyframe allows to access map points efficiently. Finally, the red-black trees of the multi-index container provide fast dereferencing of covisible keyframes, as described in Section 8.2.3.

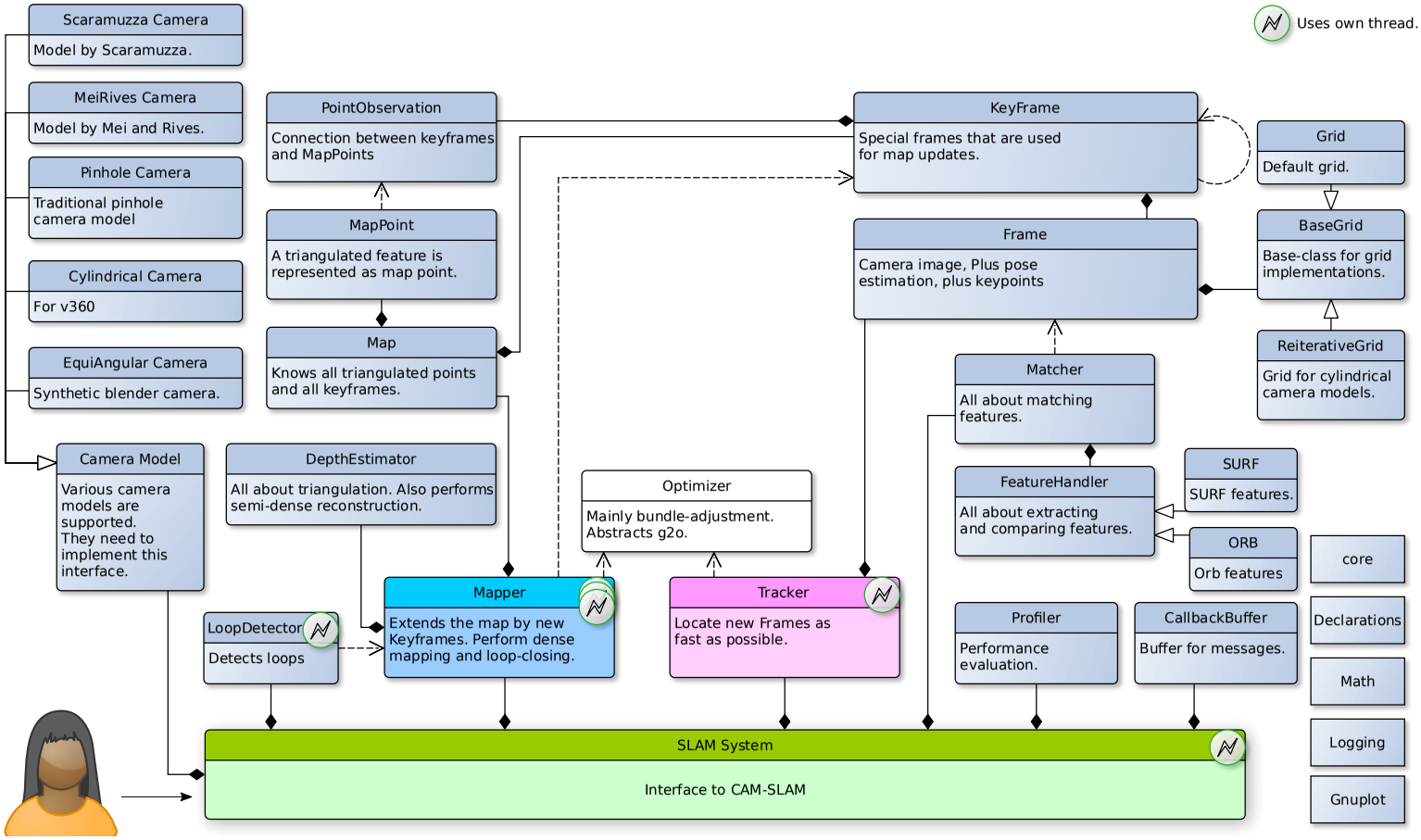


Figure A.5: Illustrative class diagram of CAM-SLAM. Classes related to camera models are placed on the left hand side, and continuing to the middle are mapping-related classes. On the right hand side, one can find frame, keyframe and matching related classes. The most dominant parts of the system, namely the tracking, mapping and system class, are placed at the bottom. Static or global structures are aligned to the bottom-right.

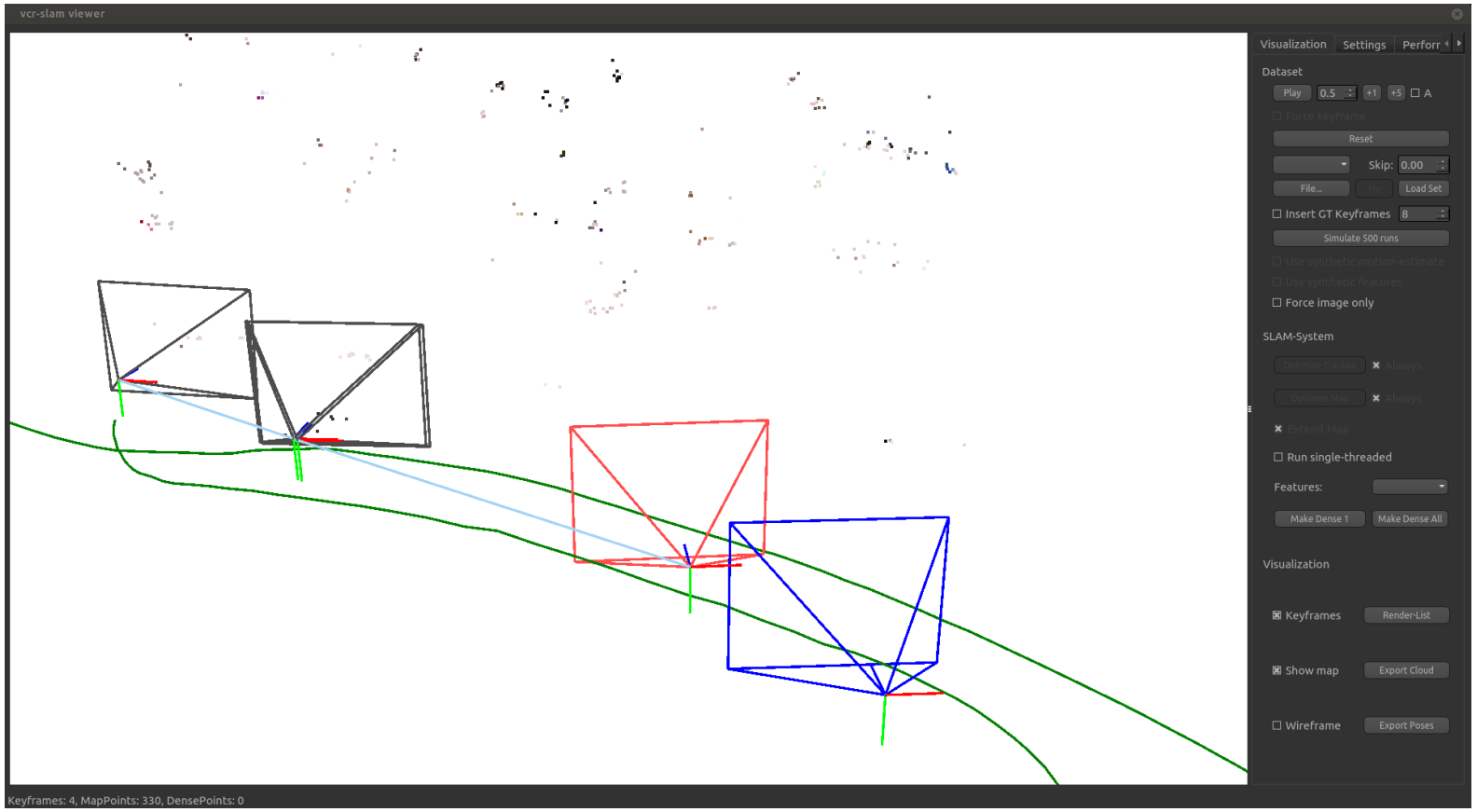


Figure A.6: Main window of the *cam-slam-viewer*. The predominant interface widget is the 3D renderer, which shows 4 keyframes (3 in grey and the selected one in red), the current frame in blue and a set of map points. Note that two of the keyframes are created in quick succession and nearly coincide. This behavior allows the spawning of more map points after initialization. Additionaly, the ground-truth camera path is highlighted in green and keyframes are connected by a blue line.

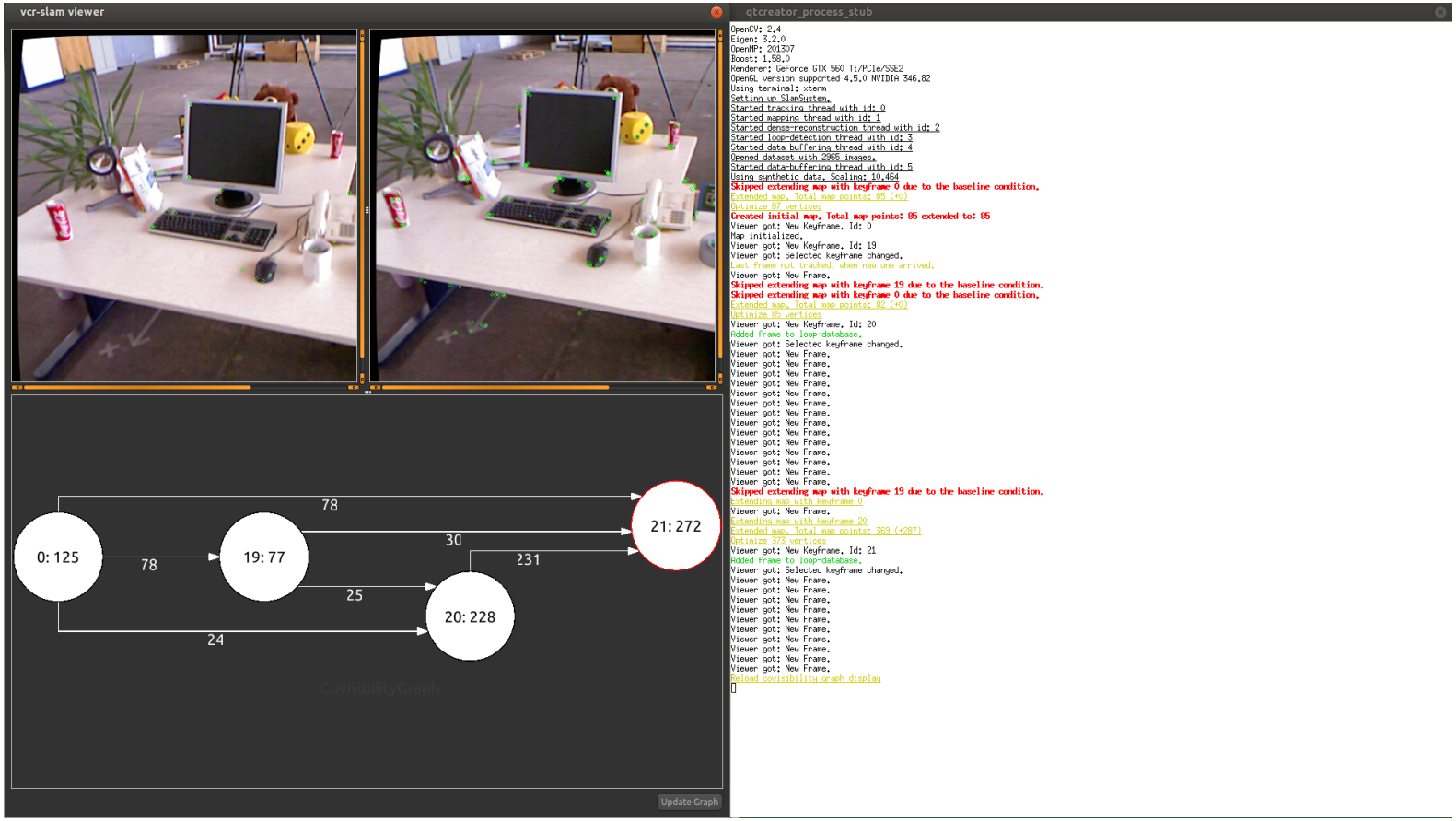


Figure A.7: Second screen of the *cam-slam-viewer*. While system output is shown on the right side, the image of the last keyframe and current frame are depicted on the top left side. The black border around the images derives from distortion correction. On the bottom-left side, the covisibility graph is visualized.

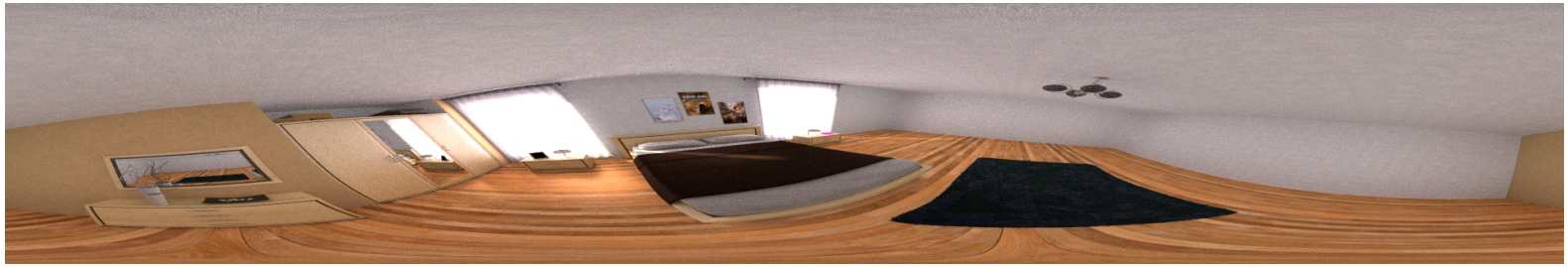


Figure A.8: Single frame of the synthetic omnidirectional room dataset which was used during experiments.

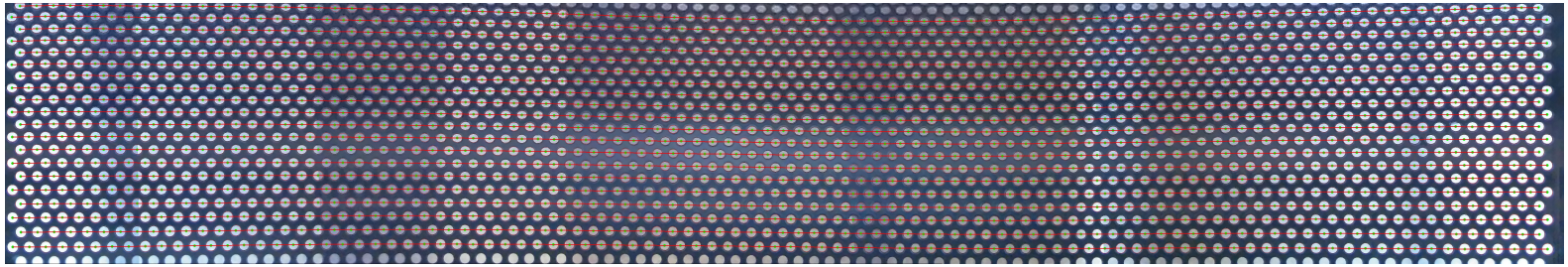


Figure A.9: Picture shot with the V.360° from within a metal tube with a structured pattern of round holes. Detected midpoints were compared to projected 3D point in order to perform camera calibration.

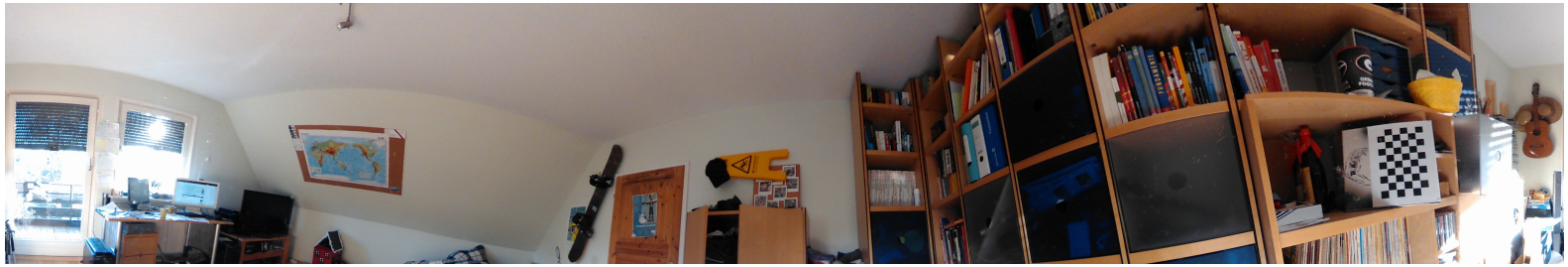


Figure A.10: Complete picture shot with the V.360° camera at a resolution of 6480 × 1080. On a closer inspection, it is possible to observe artifacts resulting from post-processing steps that are performed on the camera.

Appendix B

Additional Listings

```
typedef std::function<void(KeyframePointerConst)>  KeyframeListener;
typedef std::function<void(FramePointerConst)>  FrameListener;
typedef std::function<void(MapPointerConst)>  MapListener;

// Listeners
void addNewKeyframeListener(const KeyframeListener& listener);
void addDenseKeyframeListener(const KeyframeListener& listener);
void addSelectedKeyframeChangedListener(const KeyframeListener&
                                        listener);
void addNewFrameListener(const FrameListener& listener);
void addSelectedMapFragmentListener(const MapListener& listener);

 /// Setup SlamSystem, using a specific feature Handler
void setup(shared_ptr<FeatureHandler> featureHandler =
           std::make_shared<ORBHandler>(),
           bool enableAutoDense = false,
           bool loopDetection = false);

/// Set default camera model
void setDefaultCameraModel(CameraModelPointer model);

/// Start all sub-systems (threads), wait for data
void start();

/// Shutdown sub-systems (threads)
void shutdown();

/// Usually used for synthetic datasets.
/// Allows the user to create the frame first and then run as normal.
void addFrame(FramePointer frame);
```

```
/// Add image, make frame / keyframe,
/// using the default camera model, then call addFrame
void createFrame(cv::Mat image);

/// Add image, make frame / keyframe, then call addFrame
void createFrame(cv::Mat image, CameraModelPointer cameraModel);

/// Intrsuct the system to create a semi-dense (inverse)
/// depth map for either all or the next KF
void reconstructDense(bool all=false);

/// Get last tracked frame
FramePointerConst getLastFrameTracked() const;

/// Get last tracked frame
FramePointerConst getLastFrameWithKeypoints() const;

/// Get last added frame
FramePointerConst getLastFrame() const;

/// Get selected keyframe
KeyframePointerConst getSelectedKeyframe() const;

/// Return the map
MapPointerConst getMap() const;

/// Count mapped keyframes
size_t countKeyframes() const;

/// Count map points
size_t countMapPoints() const;

/// Is the system shutting down?
bool isShuttingDown() const;

/// Is tracking good? Returns false when the last frame wasn't
/// tracked properly, even when the tracking is still active.
bool isTrackingGood() const;
```

Listing B.1: Public interface of CAM-SLAM. The entire functionality of CAM-SLAM is encapsulated by this easy to use interface.

```
do{
    // skip when segment too small
    if(cameraModel->getDistance(start, end) < tMinLength){
        // continue with stack
        if(stack.empty()) break;
        start = end;
        end = stack.pop();
        continue;
    }

    // sample inverse depth-range
    Vec2 mid = subdivide(start, end);
    if(!cameraModel->isValidPixel(mid)) break;

    // check condition
    Vec2 sm = (mid-start).normalized();
    Vec2 me = (end-mid).normalized();
    if(sm.dot(me) < tMinDot){
        // Subdivide first half and add second one to stack
        stack.push(end);
        end = mid;
        continue;
    }

    // epipolar line sampling
    sampleLine(start, end);

    // continue with stack
    if(stack.empty()) break;
    start = end;
    end = stack.pop();
}while(true);
```

Listing B.2: C++-like pseudo-code for the line-simplification algorithm used in CAM-SLAM. Whenever a segment is sub-sampled, the second half is stored on a stack, while the first one is processed immediately.

List of Tables

List of Figures

Bibliography

[ABO01] Argyros, A.A. ; Bekris, K.E. ; Orphanoudakis, S.C.: Robot homing based on corner tracking in a sequence of panoramic images. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* Bd. 2, 2001. – ISSN 1063–6919, S. II–3–II–10 vol.2

[AF07] Arican, Zafer ; Frossard, Pascal: Dense disparity estimation from omnidirectional images. In: *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, IEEE, 2007, 399–404

[AHB87] Arun, K.S. ; Huang, T.S. ; Blostein, S.D.: Least-Squares Fitting of Two 3-D Point Sets. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-9 (1987), Sept, Nr. 5, S. 698–700. http://dx.doi.org/10.1109/TPAMI.1987.4767965. – DOI 10.1109/TPAMI.1987.4767965. – ISSN 0162–8828

[ASSS10] Agarwal, Sameer ; Snavely, Noah ; Seitz, Steven M. ; Szeliski, Richard: Bundle adjustment in the large. Version: 2010. http://link.springer.com/chapter/10.1007/978-3-642-15552-9_3. In: *Computer Vision–ECCV 2010*. Springer, 2010, 29–42

[ATA13] Aliakbarpour, H. ; Tahri, O. ; Araujo, H.: Image-based servoing of non-holonomic vehicles using non-central catadioptric cameras. In: *Robot Motion and Control (RoMoCo), 2013 9th Workshop on*, 2013, S. 54–59

[BDC01] Broadhurst, Adrian ; Drummond, Tom W. ; Cipolla, Roberto: A probabilistic framework for space carving. In: *Com-*

puter Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on Bd. 1, IEEE, 2001, 388–393

[BG14] Banissi, E. ; Golipour, M.K.: A New General Incremental Algorithm for Conic Section. In: *Computer Graphics, Imaging and Visualization (CGIV), 2014 11th International Conference on*, 2014, S. 46–51

[BJr07] Byröd, Martin ; Josephson, Klas ; Åström, Kalle: Fast Optimal Three View Triangulation. Version: 2007. http://dx.doi.org/10.1007/978-3-540-76390-1_54. In: Yagi, Yasushi (Hrsg.) ; Kang, SingBing (Hrsg.) ; Kweon, InSo (Hrsg.) ; Zha, Hongbin (Hrsg.): *Computer Vision – ACCV 2007* Bd. 4844. Springer Berlin Heidelberg, 2007. – ISBN 978–3–540–76389–5, 549-559

[BK01a] Benosman, R. ; Kang, S.B.: *Panoramic Vision: Sensors, Theory, and Applications*. Springer, 2001 (Monographs in Computer Science). https://books.google.de/books?id=MoAvSTApdDoC. – ISBN 9780387951119

[BK01b] Bunschoten, Roland ; Kröse, Ben: Range estimation from a pair of omnidirectional images. In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* Bd. 2, 2001. – ISSN 1050–4729, S. 1174–1179 vol.2

[BK03] Bunschoten, Roland ; Kröse, Ben: Robust scene reconstruction from an omnidirectional vision system. In: *Robotics and Automation, IEEE Transactions on* 19 (2003), Apr, Nr. 2, S. 351–357. http://dx.doi.org/10.1109/TRA.2003.808850. – DOI 10.1109/TRA.2003.808850. – ISSN 1042–296X

[BKDV08] Bazin, J.-C. ; Kweon, Inso ; Demonceaux, C. ; Vasseur, P.: Automatic calibration of catadioptric cameras in urban environment. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 2008, S. 3108–3114

[BM04] Baker, Simon ; Matthews, Iain: Lucas-kanade 20 years on: A unifying framework: Part 1. In: *International journal of computer vision* 56 (2004), Nr. 3, 221–255. http://link.springer.com/article/10.1023/B:VISI.0000011205.11775.fd

[BN98] Baker, Simon ; Nayar, Shree K.: A theory of catadioptric image formation. In: *Computer Vision, 1998. Sixth International Conference on*, IEEE, 1998, 35–42

[BN99] Baker, Simon ; Nayar, Shree K.: A theory of single-viewpoint catadioptric image formation. In: *International Journal of Computer Vision* 35 (1999), Nr. 2, 175–196. http://link.springer.com/article/10.1023/A:1008128724364

[BPA03] Bunschoten, Roland ; Prof, Commissie ; Adriaans, P. W.: *Mapping and Localization from a Panoramic Vision Sensor*, University of Amsterdam, Diss., 2003

[Bra00] Bradski, G.: The OpenCV Library. In: *Dr. Dobb's Journal of Software Tools* (2000)

[Bre65] Bresenham, J. E.: Algorithm for Computer Control of a Digital Plotter. In: *IBM Syst. J.* 4 (1965), März, Nr. 1, 25–30. http://dx.doi.org/10.1147/sj.41.0025. – DOI 10.1147/sj.41.0025. – ISSN 0018–8670

[BSCN08] Burbridge, Chris ; Spacek, Libor ; Condell, Joan ; Nehmzow, Ulrich: Monocular Omnidirectional Vision based Robot Localisation and Mapping. In: *Proc. of the TAROS* (2008). http://www.researchgate.net/profile/Chris_Burbridge/publication/228849878_Monocular_Omnidirectional_Vision_based_Robot_Localisation_and_Mapping/links/53ec9ebb0cf250c8947cd425.pdf

[BTVG06] Bay, Herbert ; Tuytelaars, Tinne ; Van Gool, Luc: Surf: Speeded up robust features. Version: 2006. http://link.springer.com/chapter/10.1007/11744023_32. In: *Computer vision–ECCV 2006*. Springer, 2006, 404–417

[CDM08] Civera, J. ; Davison, A.J. ; Montiel, J.: Inverse Depth Parametrization for Monocular SLAM. In: *Robotics, IEEE Transactions on* 24 (2008), Oct, Nr. 5, S. 932–945. http://dx.doi.org/10.1109/TRO.2008.2003276. – DOI 10.1109/TRO.2008.2003276. – ISSN 1552–3098

[CEC15] Caruso, D. ; Engel, J. ; Cremers, D.: Large-Scale Direct SLAM for Omnidirectional Cameras. In: *International Conference on Intelligent Robots and Systems (IROS)*, 2015

[CH15] Cortesi, Aldo ; Hils, Maximilian: *mitmproxy - home*. https://mitmproxy.org/, 2015. – [Online; accessed 07-September-2015]

[CLSF10] Calonder, Michael ; Lepetit, Vincent ; Strecha, Christoph ; Fua, Pascal: Brief: Binary robust independent elementary features. In: *Computer Vision–ECCV 2010* (2010), 778–792. http://www.springerlink.com/index/h8h1824827036042.pdf

[CN07] Cummins, Mark ; Newman, Paul: Probabilistic appearance based navigation and loop closing. In: *Robotics and automation, 2007 IEEE international conference on*, IEEE, 2007, 2042–2048

[CN08] Cummins, M. ; Newman, P.: FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. In: *The International Journal of Robotics Research* 27 (2008), Juni, Nr. 6, 647–665. http://dx.doi.org/10.1177/0278364908090961. – DOI 10.1177/0278364908090961. – ISSN 0278–3649

[Com15] Company, The Q.: *Qt - Home*. http://www.qt.io/developers/, 2015. – [Online; accessed 05-September-2015]

[Cor11] Corke, P.: *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011 (Springer Tracts in Advanced Robotics). https://books.google.com.au/books?id=hdkytqtBcyQC. – ISBN 9783642201431

[CS99] Chahl, J. S. ; Srinivasan, M. V.: Panoramic range estimation using a moving camera and a specially shaped reflective surface. In: *Proceedings of the Australian Conference on Robotics and Automation*, 1999, S. 246–251

[Dav03] Davison, Andrew J.: Real-time simultaneous localisation and mapping with a single camera. In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, IEEE, 2003, 1403–1410

[DM98] Dagum, Leonardo ; Menon, Ramesh: OpenMP: an industry standard API for shared-memory programming. In: *Computational Science & Engineering, IEEE* 5 (1998), Nr. 1, S. 46–55

[DRMS07] Davison, Andrew J. ; Reid, Ian D. ; Molton, Nicholas D. ; Stasse, Olivier: MonoSLAM: Real-Time Single Camera

SLAM. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (2007), Juni, Nr. 6, 1052–1067. http://dx.doi.org/10.1109/TPAMI.2007.1049. – DOI 10.1109/TPAMI.2007.1049. – ISSN 0162–8828

[DS96] Dennis, J. E. Jr. ; Schnabel, Robert B.: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Classics in Applied Mathematics, 16)*. Soc for Industrial & Applied Math, 1996. – 155–159 S. – ISBN 0898713641

[ED06] Eade, Ethan ; Drummond, Tom: Scalable monocular SLAM. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* Bd. 1, IEEE, 2006, 469–476

[EGK+01] Ellson, John ; Gansner, Emden ; Koutsofios, Lefteris ; North, Stephen ; Woodhull, Gordon ; Description, Short ; Technologies, Lucent: Graphviz — open source graph drawing tools. In: *Lecture Notes in Computer Science*, Springer-Verlag, 2001, S. 483–484

[Eic15] Eichhammer, Emanuel: *Qt Plotting Widget QCustomPlot - Introduction*. http://www.qcustomplot.com/, 2015. – [Online; accessed 05-September-2015]

[ESC13] Engel, Jakob ; Sturm, Jurgen ; Cremers, Daniel: Semi-Dense Visual Odometry for a Monocular Camera. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*, IEEE, 2013, 1449–1456

[ESC14] Engel, J. ; Schöps, T. ; Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In: *European Conference on Computer Vision (ECCV)*, 2014

[ESN06] Engels, Chris ; Stewénius, Henrik ; Nistér, David: Bundle adjustment rules. In: *In Photogrammetric Computer Vision*, 2006

[FAH71] Fletcher, R. ; Authority, United Kingdom Atomic E. ; H.M.S.O.: *A Modified Marquardt Subroutine for Non-linear Least Squares*. Theoretical Physics Division, Atomic Energy Research Establishment, 1971 (AERE report / R: AERE report). https://books.google.de/books?id=YajHHAAACAAJ

[FB81] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, 381–395. http://dl.acm.org/citation.cfm?id=358692

[FBB+05] Fleck, S. ; Busch, F. ; Biber, P. ; Strasser, W. ; Andreasson, H.: Omnidirectional 3D Modeling on a Mobile Robot using Graph Cuts. In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, S. 1748–1754

[Fou15] Foundation, Stichting B.: *Home of the Blender project - Free and Open 3D Creation Software*. http://www.blender.org/, 2015. – [Online; accessed 06-September-2015]

[FP02] Forsyth, David A. ; Ponce, Jean: *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002. – ISBN 0130851981

[FPS14] Forster, C. ; Pizzoli, M. ; Scaramuzza, D.: SVO: Fast semi-direct monocular visual odometry. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, S. 15–22

[FSMB98] Franz, Matthias O. ; Schölkopf, Bernhard ; Mallot, Hanspeter A. ; Bülthoff, Heinrich H.: Where did I take that snapshot? Scene-based Homing by Image Matching. In: *Biological Cybernetics* 79 (1998), S. 191–202

[FZ98] Fitzgibbon, A. W. ; Zisserman, A.: Automatic Camera Recovery for Closed or Open Image Sequences. In: *European Conference on Computer Vision*, Springer-Verlag, 1998, S. 311–326

[Gal11] Gallier, Jean: *Texts in Applied Mathematics*. Bd. 38: *Geometric Methods and Applications*. New York, NY : Springer New York, 2011 http://link.springer.com/10.1007/978-1-4419-9961-0. – ISBN 978–1–4419–9960–3, 978–1–4419–9961–0

[GD99] Geyer, Christopher ; Daniilidis, Konstantinos: Catadioptric camera calibration. In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on* Bd. 1, IEEE, 1999, 398–404

[GD00] Geyer, Christopher ; Daniilidis, Kostas: A unifying theory for central panoramic systems and practical implications. Version: 2000. http://link.springer.com/chapter/10.1007/3-540-45053-X_29. In: *Computer Vision—ECCV 2000*. Springer, 2000, 445–461

[GD01] Geyer, Christopher ; Daniilidis, Kostas: Catadioptric projective geometry. In: *International Journal of Computer Vision* 45 (2001), Nr. 3, 223–243. http://link.springer.com/article/10.1023/A:1013610201135

[Geb03] Geb, Thomas: Real-time panospheric image dewarping and presentation for remote mobile robot control. In: *Advanced Robotics* 17 (2003), Nr. 4, 359-368. http://dx.doi.org/10.1163/156855303765203047. – DOI 10.1163/156855303765203047

[GJ+10] Guennebaud, Gaël ; Jacob, Benoît u. a.: *Eigen v3*. http://eigen.tuxfamily.org, 2010

[GJMSMCMJ14] Garrido-Jurado, S. ; Muñoz-Salinas, R. ; Madrid-Cuevas, F.J. ; Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. In: *Pattern Recognition* 47 (2014), Nr. 6, 2280 - 2292. http://dx.doi.org/http://dx.doi.org/10.1016/j.patcog.2014.01.005. – DOI http://dx.doi.org/10.1016/j.patcog.2014.01.005. – ISSN 0031–3203

[GKSK11] Grisetti, Giorgio ; Kümmerle, Rainer ; Strasdat, Hauke ; Konolige, Kurt: g2o: A general Framework for (Hyper) Graph Optimization / Technical report. Version: 2011. http://openslam.informatik.uni-freiburg.de/data/svn/g2o/trunk/doc/g2o.pdf. 2011. – Forschungsbericht

[GLT12] Gálvez-López, D. ; Tardos, J.D.: Bags of Binary Words for Fast Place Recognition in Image Sequences. In: *Robotics, IEEE Transactions on* 28 (2012), Oct, Nr. 5, S. 1188–1197. – ISSN 1552–3098

[GLU12] Geiger, Andreas ; Lenz, Philip ; Urtasun, Raquel: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012

[GMR13] Gamallo, C. ; Mucientes, Manuel ; Regueiro, Carlos V.: A FastSLAM-based Algorithm for Omnidirectional Cameras. In: *Journal of Physical Agents* 7 (2013), Nr. 1

[GMW+12] Glover, A. ; Maddern, W. ; Warren, M. ; Reid, S. ; Milford, M. ; Wyeth, Gordon: OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012. – ISSN 1050–4729, S. 4730–4735

[GN98] Gluckman, Joshua ; Nayar, Shree K.: Ego-motion and omnidirectional cameras. In: *Computer Vision, 1998. Sixth International Conference on*, IEEE, 1998, 999–1005

[GNT98] Gluckman, Joshua ; Nayar, Shree K. ; Thoresz, Keith J.: Real-time omnidirectional and panoramic stereo. In: *Proc. of Image Understanding Workshop* Bd. 1, Citeseer, 1998, 299–303

[GNTVG07] Goedemé, Toon ; Nuttin, Marnix ; Tuytelaars, Tinne ; Van Gool, Luc: Omnidirectional vision based topological navigation. In: *International Journal of Computer Vision* 74 (2007), Nr. 3, 219–236. http://link.springer.com/article/10.1007/s11263-006-0025-9

[GRMG11] Gutierrez, Daniel ; Rituerto, Alejandro ; Montiel, J. M. M. ; Guerrero, José J.: Adapting a real-time monocular visual slam from conventional to omnidirectional cameras. In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, IEEE, 2011, 343–350

[GS07] Geyer, Christopher ; Stewenius, Henrik: A nine-point algorithm for estimating para-catadioptric fundamental matrices. In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, IEEE, 2007, 1–8

[GS08] Gebken, Christian ; Sommer, Gerald: Stochastically optimal epipole estimation in omnidirectional images with geometric algebra. Version: 2008. http://link.springer.com/chapter/10.1007/978-3-540-78157-8_7. In: *Robot Vision*. Springer, 2008, 85–97

[GTVG+05] Goedemé, Toon ; Tuytelaars, Tinne ; Van Gool, Luc ; Vanacker, Gerolf ; Nuttin, Marnix: Feature based omnidirectional sparse visual path following. In: *Intelligent Robots*

and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on, IEEE, 2005, 1806–1811

[GYK+11] Goto, Shinichi ; Yamashita, Atsushi ; Kawanishi, Ryosuke ; Kaneko, Toru ; Asama, Hajime: 3D environment measurement using binocular stereo and motion stereo by mobile robot with omnidirectional stereo camera. In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, IEEE, 2011, 296–303

[Har87] Harris, C. G.: Determination of Ego-Motion from Matched Points. In: *In Third Alvey Vision Conference*, Alvey Vision Club, 1987, 26.1–26.4

[HB01] Hicks, R. A. ; Bajcsy, R.: Reflective Surfaces as computational sensors. In: *Image and Vision Computing* 19 (2001), S. 773–777

[Her08] Hertzberg, Christoph: *A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds*. 2008

[HFRJ14] Hess-Flores, Mauricio ; Recker, Shawn ; Joy, Kenneth I.: Uncertainty, Baseline, and Noise Analysis for L1 Error-Based Multi-View Triangulation. In: *Pattern Recognition (ICPR), 2014 22nd International Conference on*, IEEE, 2014, 4074–4079

[HGC92] Hartley, R. ; Gupta, R. ; Chang, T.: Stereo from uncalibrated cameras. In: *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Comput. Soc. Press, 1992. – ISBN 0–8186–2855–3, 761–764

[HHN88] Horn, Berthold K. ; Hilden, Hugh M. ; Negahdaripour, Shahriar: Closed-form solution of absolute orientation using orthonormal matrices. In: *JOSA A* 5 (1988), Nr. 7, 1127–1135. `http://www.opticsinfobase.org/abstract.cfm?uri=josaa-5-7-1127`

[HKS08] Huang, F. ; Klette, R. ; Scheibe, K.: *Panoramic Imaging: Sensor-Line Cameras and Laser Range-Finders*. Wiley, 2008 (The Wiley-IS&T Series in Imaging Science and Technology). `https://books.google.de/books?id=75xwJ_yiCc4C`. – ISBN 9780470998274

[HLP13] Heng, Lionel ; Li, Bo ; Pollefeys, Marc: CamOdoCal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In: *IROS*, IEEE, 2013, 1793-1800

[HP87] Harris, C. G. ; Pike, J. M.: 3D Positional Integration from Image Sequences. In: *In Proc. Alvey Vision Conference, Cambridge.England*, Alvey Vision Club, 1987, 32.1–32.4

[HS97] Hartley, Richard I. ; Sturm, Peter: Triangulation. In: *Computer vision and image understanding* 68 (1997), Nr. 2, S. 146–157

[HS09] Hirschmüller, Heiko ; Scharstein, Daniel: Evaluation of stereo matching costs on images with radiometric differences. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 31 (2009), Nr. 9, 1582–1599. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4620119

[HWFS11] Hertzberg, Christoph ; Wagner, René ; Frese, Udo ; Schröder, Lutz: Integrating Generic Sensor Fusion Algorithms with Sound State Representations through Encapsulation of Manifolds. In: *CoRR* abs/1107.1119 (2011). http://arxiv.org/abs/1107.1119

[HZ04] Hartley, R. I. ; Zisserman, A.: *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004

[Inc15] Inc., Google: *googletest - Google C++ Testing Framework*. https://code.google.com/p/googletest/, 2015. – [Online; accessed 05-September-2015]

[Ish98] Ishiguro, Hiroshi: Development of low-cost compact omnidirectional vision sensors and their applications. In: *Proc. Int. Conf. Information systems, analysis and synthesis*, 1998, 433–439

[IYT92] Ishiguro, H. ; Yamamoto, M. ; Tsuji, S.: Omni-directional stereo. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 14 (1992), Feb, Nr. 2, S. 257–262. http://dx.doi.org/10.1109/34.121792. – DOI 10.1109/34.121792. – ISSN 0162–8828

[JNS+10] Jeong, Yekeun ; Nister, D. ; Steedly, D. ; Szeliski, R. ; Kweon, In-So: Pushing the envelope of modern methods for bundle adjustment. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010. – ISSN 1063–6919, S. 1474–1481

[KJLS14] Kim, JeongWoon ; Jung, Yeondeuk ; Lee, Dasol ; Shim, David H.: Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In: *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, IEEE, 2014, 1243–1252

[KM09] Klein, Georg ; Murray, David: Parallel Tracking and Mapping on a Camera Phone. In: *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*. Washington, DC, USA : IEEE Computer Society, 2009 (ISMAR '09). – ISBN 978–1–4244–5390–0, 83–86

[KM13] Klein, Georg ; Murray, David: *PTAM (Parallel Tracking and Mapping) re-released under GPLv3.* `https://github.com/Oxford-PTAM/PTAM-GPL`, 2013. – [Online; accessed 07-August-2015]

[KPB13] Kukelova, Z. ; Pajdla, T. ; Bujnak, M.: Fast and Stable Algebraic Solution to L2 Three-View Triangulation. In: *3D Vision - 3DV 2013, 2013 International Conference on*, 2013, S. 326–333

[KS97] Kang, Sing B. ; Szeliski, Richard: 3-D scene data recovery using omnidirectional multibaseline stereo. In: *International Journal of Computer Vision* 25 (1997), Nr. 2, 167–183. `http://link.springer.com/article/10.1023/A:1007971901577`

[KS00] Kutulakos, Kiriakos N. ; Seitz, Steven M.: A theory of shape by space carving. In: *International Journal of Computer Vision* 38 (2000), Nr. 3, 199–218. `http://link.springer.com/article/10.1023/A:1008191222954`

[KSC13] Kerl, C. ; Sturm, J. ; Cremers, D.: Dense Visual SLAM for RGB-D Cameras. In: *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013

[KSN08] Kanatani, Kenichi ; Sugaya, Yasuyuki ; Niitsuma, Hirotaka: Triangulation from two views revisited: Hartley-Sturm vs. optimal correction. In: *In practice* 4 (2008), 5. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.220.724&rep=rep1&type=pdf

[Ku66] Ku, H. H.: Notes on the use of propagation of error formulas. In: *Journal of Research of the National Bureau of Standards. Section C: Engineering and Instrumentation* 70C (1966), Oktober, Nr. 4, S. 263–273. – ISSN 0022–4316

[KYK11] Kawanishi, Ryosuke ; Yamashita, Atsushi ; Kaneko, Toru: *Three-Dimensional Environment Modeling Based on Structure from Motion with Point and Line Features by Using Omnidirectional Camera.* INTECH Open Access Publisher, 2011 http://cdn.intechweb.org/pdfs/24916.pdf

[KYKA12] Kawanishi, R. ; Yamashita, A. ; Kaneko, T. ; Asama, H.: Line-based camera movement estimation by using parallel lines in omnidirectional video. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012. – ISSN 1050–4729, S. 3573–3579

[Lab06] Labrosse, Frédéric: The visual compass: Performance and limitations of an appearance-based method. In: *Journal of Field Robotics* 23 (2006), Oktober, Nr. 10, 913–941. http://dx.doi.org/10.1002/rob.20159. – DOI 10.1002/rob.20159. – ISSN 15564959, 15564967

[LAo05] Lourakis, Manolis I. ; Argyros, Antonis ; others: Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* Bd. 2, IEEE, 2005, 1526–1531

[Lev44] Levenberg, K.: A method for the solution of certain problems in least squares. In: *Quart. Applied Math.* 2 (1944), S. 164–168

[LH06] Li, Hongdong ; Hartley, Richard: Five-point motion estimation made easy. In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on* Bd. 1, IEEE, 2006, 630–633

[LHS+07] Luo, Chuanjiang ; He, Lei ; Su, Liancheng ; Zhu, Feng ; Hao, Yingming ; Shi, Zelin: Omnidirectional depth recovery based on a novel stereo sensor. In: *Image* 2 (2007), Nr. u2, S. v2

[Lin10] Lindstrom, Peter: Triangulation made easy. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, 2010, 1554–1561

[LK81] Lucas, Bruce D. ; Kanade, Takeo: An Iterative Image Registration Technique with an Application to Stereo Vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1981 (IJCAI'81), 674–679

[LM97] Lu, F. ; Milios, E.: Globally Consistent Range Scan Alignment for Environment Mapping. In: *AUTONOMOUS ROBOTS* 4 (1997), S. 333–349

[Lon81] Longuet: A computer algorithm for reconstructing a scene from two projections. In: *Nature* 293 (1981), September, S. 133–135

[Low04] Lowe, David G.: Distinctive image features from scale-invariant keypoints. In: *International journal of computer vision* 60 (2004), Nr. 2, 91–110. http://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94

[MA15] Mur-Artal, Raúl: *ORBextractor.h*. https://github.com/raulmur/ORB_SLAM/blob/master/include/ORBextractor.h, 2015. – [Online; accessed 05-September-2015]

[MAMT15] Mur-Artal, Raúl ; Montiel, J. M. M. ; Tardós, Juan D.: ORB-SLAM: a Versatile and Accurate Monocular SLAM System. In: *Submitted to IEEE Transaction on Robotics. arXiv preprint arXiv:1502.00956* (2015)

[Mar63] Marquardt, Donald W.: An algorithm for least-squares estimation of nonlinear parameters. In: *SIAM Journal on Applied Mathematics* 11 (1963), Nr. 2, 431–441. http://dx.doi.org/10.1137/0111030. – DOI 10.1137/0111030

[MAT14] Mur-Artal, R. ; Tardos, J.D.: Fast relocalisation and loop closing in keyframe-based SLAM. In: *Robotics and Automation*

(ICRA), 2014 IEEE International Conference on, 2014, S. 846–853

[MAT15] Mur-Artal, Raul ; Tardós, Juan D.: Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM. In: *Robotics: Science and Systems*, 2015

[MB95] McMillan, Leonard ; Bishop, Gary: Plenoptic modeling: An image-based rendering system. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 1995, 39–46

[MC11] Matas, J. ; Chum, O.: *RANSAC in 2011 (30 years after).* http://www.imgfsr.com/CVPR2011/Tutorial6/RANSAC_CVPR2011.pdf. Version: 2011. – Presentation about RANSAC at CVPR 2011. [Accessed: 2015 07 16]

[MGS07] Murillo, Ana C. ; Guerrero, José J. ; Sagues, C.: Surf features for efficient robot localization with omnidirectional images. In: *Robotics and Automation, 2007 IEEE International Conference on*, IEEE, 2007, 3901–3907

[Mic04] Micušik, Branislav: *Two-view geometry of omnidirectional cameras*, Czech Technical University in Prague, Diss., 2004. http://cmp.felk.cvut.cz/ftp/articles/micusik/Micusik-thesis.pdf

[ML09] Muja, Marius ; Lowe, David G.: Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In: *International Conference on Computer Vision Theory and Application VISSAPP'09)*, INSTICC Press, 2009, S. 331–340

[Mob15] Mobil, VSN: *VSN Mobil Support.* https://support.vsnmobil.com/hc/en-us,https://p4.zdassets.com/hc/theme_assets/544440/200026195/v360.gif, 2015. – [Online; accessed 10-September-2015]

[MP03] Micusik, Branislav ; Pajdla, Tomas: Estimation of omnidirectional camera model from epipolar geometry. In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* Bd. 1, IEEE, 2003, I–485

[MR07] Mei, Christopher ; Rives, Patrick: Single view point omnidirectional camera calibration from planar grids. In: *Robotics and*

Automation, 2007 IEEE International Conference on, IEEE, 2007, 3945–3950

[MSN10] Mei, Christopher ; Sibley, Gabe ; Newman, Paul: Closing loops without places. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, 3738–3744

[Mun04] Munoz, Joaquin M L.: The Boost Multi-Index Containers Library. In: *C/C++ Users Journal* Vol. 22, Issue 9 (2004), September, S. 6

[MWB06] Mrazek, P. ; Weickert, J. ; Bruhn, A.: On Robust Estimation and smoothing with Spatial and Tonal Kernels. Version: 2006. `http://dx.doi.org/10.1007/1-4020-3858-8_18`. In: Klette, Reinhard (Hrsg.) ; Kozera, Ryszard (Hrsg.) ; Noakes, Lyle (Hrsg.) ; Weickert, Joachim (Hrsg.): *Geometric Properties for Incomplete data* Bd. 31. Springer Netherlands, 2006. – DOI 10.1007/1–4020–3858–8_18. – ISBN 978–1–4020–3857–0, 335-352

[Nay88] Nayar, Shree K.: Sphereo: Determining Depth using Two Specular Spheres and a Single Camera. In: *1988 Cambridge Symposium on Advances in Intelligent Robotics Systems*, 1988

[Nay97] Nayar, S.K.: Catadioptric omnidirectional camera. In: *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, 1997. – ISSN 1063–6919, S. 482–488

[Nic14] Nicolas, Bergont: *Interactive Qt graphViz display.* `https://github.com/nbergont/qgv`, 2014. – [Online; accessed 05-September-2015]

[NLD11] Newcombe, Richard A. ; Lovegrove, Steven J. ; Davison, Andrew J.: DTAM: Dense tracking and mapping in real-time. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*, IEEE, 2011, 2320–2327

[NN91] Nocedal, Jorge ; Nash, Stephen G.: A Numerical Study of the Limited Memory BFGS Method and the Truncated-Newton Method for Large Scale Optimization. In: *SIAM Journal on Optimization* 1 (1991), Nr. 3, S. 358–372

[Nor08] Nordberg, Klas: Efficient triangulation based on 3d euclidean optimization. In: *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, IEEE, 2008, 1–4

[NW06] Nocedal, Jorge ; Wright, Stephen J.: *Numerical Optimization, second edition*. World Scientific, 2006

[OH87] Oh, Sung J. ; Hall, Ernest L.: Guidance Of A Mobile Robot Using An Omnidirectional Vision Navigation System. In: *Robotics and IECON'87 Conferences* Bd. 0852 International Society for Optics and Photonics, 1987, 288-300

[OLT06] Olson, Edwin ; Leonard, John ; Teller, Seth: Fast iterative alignment of pose graphs with poor initial estimates. In: *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, IEEE, 2006, 2262–2269

[PFS14] Pizzoli, Matia ; Forster, Christian ; Scaramuzza, Davide: REMODE: Probabilistic, monocular dense reconstruction in real time. In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, 2609–2616

[PN97] Peri, Venkata ; Nayar, Shree K.: Generation of perspective and panoramic video from omnidirectional video. In: *Proc. DARPA Image Understanding Workshop* Bd. 1, Citeseer, 1997, 243–245

[PNF+08] Pollefeys, Marc ; Nistér, David ; Frahm, J.-M. ; Akbarzadeh, Amir ; Mordohai, Philippos ; Clipp, Brian ; Engels, Chris ; Gallup, David ; Kim, S.-J. ; Merrell, Paul ; others: Detailed real-time urban 3d reconstruction from video. In: *International Journal of Computer Vision* 78 (2008), Nr. 2-3, 143–167. http://link.springer.com/article/10.1007/s11263-007-0086-4

[PO13] Phan, Khoa D. ; Ovchinnikov, Aleksandr V.: Indoor Slam Using an Omnidirectional Camera. In: *Middle-East Journal of Scientific Research* 16 (2013), Nr. 1, 88–94. http://www.idosi.org/mejsr/mejsr16(1)13/13.pdf

[Pri12] Prince, Simon J. D.: *Computer Vision: Models, Learning, and Inference*. 1st. New York, NY, USA : Cambridge University Press, 2012. – ISBN 1107011795, 9781107011793

[PS11] Pagani, Alain ; Stricker, Didier: Structure from Motion using full spherical panoramic cameras. In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, IEEE, 2011, 375–382

[QCG+09] Quigley, Morgan ; Conley, Ken ; Gerkey, Brian P. ; Faust, Josh ; Foote, Tully ; Leibs, Jeremy ; Wheeler, Rob ; Ng, Andrew Y.: ROS: an open-source Robot Operating System. In: *ICRA Workshop on Open Source Software*, 2009

[Ram72] Ramer, Urs: An iterative procedure for the polygonal approximation of plane curves. In: *Computer Graphics and Image Processing* 1 (1972), November, Nr. 3, 244–256. http://dx.doi.org/10.1016/s0146-664x(72)80017-0. – DOI 10.1016/s0146–664x(72)80017–0. – ISSN 0146664X

[RD06] Rosten, Edward ; Drummond, Tom: Machine learning for high-speed corner detection. Version: 2006. http://link.springer.com/chapter/10.1007/11744023_34. In: *Computer Vision–ECCV 2006*. Springer, 2006, 430–443

[Ree70] Rees, D.W.: *Panoramic television viewing system*. http://www.google.com/patents/US3505465. Version: April 7 1970. – US Patent 3,505,465

[RHFJ13] Recker, Shawn ; Hess-Flores, Mauricio ; Joy, Kenneth I.: Statistical angular error-based triangulation for efficient and accurate multi-view scene reconstruction. In: *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*, IEEE, 2013, 68–75

[RPG10a] Rituerto, Alejandro ; Puig, Luis ; Guerrero, J. J.: Comparison of omnidirectional and conventional monocular systems for visual slam. In: *10th OMNIVIS with RSS* (2010). http://www.researchgate.net/profile/Alejandro_Rituerto/publication/229069514_Comparison_of_omnidirectional_and_conventional_monocular_systems_for_visual_SLAM/links/09e4150bccacacd7bf000000.pdf

[RPG10b] Rituerto, Alejandro ; Puig, Luis ; Guerrero, José J.: Visual slam with an omnidirectional camera. In: *Pattern Recognition (ICPR), 2010 20th International Conference on*, IEEE, 2010, 348–351

[RRKB11] Rublee, E. ; Rabaud, V. ; Konolige, K. ; Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011. – ISSN 1550–5499, S. 2564–2571

[Rü15] Rünz, Martin: *V360 Remote Control - Source Code.* https://github.com/martinruenz/v360, 2015. – [Online; accessed 12-August-2015]

[SBD04] Smadja, Laurent ; Benosman, Ryad ; Devars, Jean: Cylindrical sensor calibration using lines. In: *ICIP*, IEEE, 2004, 1851-1854

[SC86] Smith, Randall C. ; Cheeseman, Peter: On the Representation and Estimation of Spatial Uncertainly. In: *Int. J. Rob. Res.* 5 (1986), Dezember, Nr. 4, S. 56–68. http://dx.doi.org/10.1177/027836498600500404. – DOI 10.1177/027836498600500404. – ISSN 0278–3649

[Sca08] Scaramuzza, Davide: *Omnidirectional vision: from calibration to robot motion estimation*, Citeseer, Diss., 2008. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.4525&rep=rep1&type=pdf

[Sch12] Scheggi, S.: *Motion estimation algorithms for catadioptric cameras*, University of Siena, Diss., October 2012

[SDMK11] Strasdat, H. ; Davison, A.J. ; Montiel, J.M.M. ; Konolige, K.: Double window optimisation for constant time visual SLAM. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011. – ISSN 1550–5499, S. 2352–2359

[SEE+12] Sturm, J. ; Engelhard, N. ; Endres, F. ; Burgard, W. ; Cremers, D.: A Benchmark for the Evaluation of RGB-D SLAM Systems. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012

[SG11] Salazar-Garibay, Adan: *Direct self-calibration of central catadioptric omnidirectional cameras*, École Nationale Supérieure des Mines de Paris, Diss., 2011. https://pastel.archives-ouvertes.fr/pastel-00645697/

[SG14] Schoenbein, Miriam ; Geiger, Andreas: Omnidirectional 3D Reconstruction in Augmented Manhattan Worlds. In: *International Conference on Intelligent Robots and Systems*. Chicago, IL, USA : IEEE, Oktober 2014, S. 716 – 723

[SMS06a] Scaramuzza, Davide ; Martinelli, Agostino ; Siegwart, Roland: A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion. In: *Proceedings of the Fourth IEEE International Conference on Computer Vision Systems*. Washington, DC, USA : IEEE Computer Society, 2006 (ICVS '06). – ISBN 0–7695–2506–7, 45–

[SMS06b] Scaramuzza, Davide ; Martinelli, Agostino ; Siegwart, Roland: A toolbox for easily calibrating omnidirectional cameras. In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, 2006, 5695–5701

[SNS11] Siegwart, Roland ; Nourbakhsh, Illah R. ; Scaramuzza, Davide: *Introduction to Autonomous Mobile Robots*. 2nd. The MIT Press, 2011. – ISBN 0262015358, 9780262015356

[SP02] Svoboda, Tomáš ; Pajdla, Tomáš: Epipolar geometry for central catadioptric cameras. In: *International Journal of Computer Vision* 49 (2002), Nr. 1, 23–37. http://link.springer.com/article/10.1023/A:1019869530073

[Spe15] Specialist, Glenn VSN Mobil S.: *Support ticket number 433*. Personal communication, 05 2015

[SPH98] Svoboda, Tomáš ; Pajdla, Tomáš ; Hlaváč, Václav: Epipolar geometry for panoramic cameras. Version: 1998. http://dx.doi.org/10.1007/BFb0055669. In: Burkhardt, Hans (Hrsg.) ; Neumann, Bernd (Hrsg.): *Computer Vision — ECCV'98* Bd. 1406. Springer Berlin Heidelberg, 1998. – DOI 10.1007/BFb0055669. – ISBN 978–3–540–64569–6, 218-231

[SRL13] Schönbein, Miriam ; Rapp, Holger ; Lauer, Martin: Panoramic 3d reconstruction with three catadioptric cameras. Version: 2013. http://link.springer.com/chapter/10.1007/978-3-642-33926-4_32. In: *Intelligent Autonomous Systems 12*. Springer, 2013, 345–353

[SS08] Scaramuzza, D. ; Siegwart, R.: Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. In: *IEEE Transactions on Robotics* 24 (2008), Oktober, Nr. 5, 1015–1026. http://dx.doi.org/10.1109/TRO.2008.2004490. – DOI 10.1109/TRO.2008.2004490. – ISSN 1552–3098, 1941–0468

[SSG14] Schönbein, Miriam ; Strauss, Tobias ; Geiger, Andreas: Calibrating and Centering Quasi-Central Catadioptric Cameras. In: *International Conference on Robotics and Automation (ICRA)*, 2014

[SSN05] Stewenius, H. ; Schaffalitzky, F. ; Nister, D.: How hard is 3-view triangulation really? In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* Bd. 1, 2005. – ISSN 1550–5499, S. 686–693 Vol. 1

[Sti08] Stillwell, J.: *Naive Lie Theory*. Springer, 2008 (Undergraduate Texts in Mathematics). https://books.google.de/books?id=SuR5OAgxyDIC. – ISBN 9780387782157

[Str12a] Strasdat, Hauke: *Local accuracy and global consistency for efficient SLAM*, Imperial College London, Diss., 2012. http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.566392

[Str12b] Strasdat, Hauke: *Sophus (version 0.9a) C++ implementation of Lie Groups using Eigen*. https://github.com/strasdat/Sophus, 2012. – [Online; accessed 05-September-2015]

[Stu10] Sturm, Peter: Camera Models and Fundamental Concepts Used in Geometric Computer Vision. In: *Foundations and Trends® in Computer Graphics and Vision* 6 (2010), Nr. 1-2, 1–183. http://dx.doi.org/10.1561/0600000023. – DOI 10.1561/0600000023. – ISSN 1572–2740, 1572–2759

[Sue13] Suenderhauf, Niko: *OpenSeqSLAM*. https://openslam.org/openseqslam.html, 2013. – [Online; accessed 05-September-2015]

[Svo00a] Svoboda, Tomáš: *Central Panoramic Cameras Design, Geometry, Egomotion*. Prague, Czech Republic, Center for Machine Perception, Czech Technical University, PhD Thesis, April 2000. – 117 S.

[Svo00b] Svoboda, Tomáš: Evaluation, Transformation, and Parameterization of Epipolar Conics / Center for Machine Perception, Czech Technical University. 2000. – research report

[SW03] Seber, G.A.F. ; Wild, C.J.: *Nonlinear Regression*. Wiley, 2003 (Wiley Series in Probability and Statistics). – 624–627 S. – ISBN 9780471471356

[TBF05] Thrun, Sebastian ; Burgard, Wolfram ; Fox, Dieter: *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. – ISBN 0262201623

[TMHF00] Triggs, Bill ; McLauchlan, Philip F. ; Hartley, Richard I. ; Fitzgibbon, Andrew W.: Bundle adjustment—a modern synthesis. Version: 2000. http://link.springer.com/chapter/10.1007/3-540-44480-7_21. In: *Vision algorithms: theory and practice*. Springer, 2000, 298–372

[TPD08] Tardif, J.-P. ; Pavlidis, Yanis ; Daniilidis, Kostas: Monocular visual odometry in urban environments using an omnidirectional camera. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, IEEE, 2008, 2531–2538

[Tre13] Treiber, Marco A.: *Optimization for Computer Vision: An Introduction to Core Concepts and Methods*. Springer Publishing Company, Incorporated, 2013. – ISBN 1447152824, 9781447152828

[TZ97] Torr, Philip H. ; Zissermann, A.: Performance characterization of fundamental matrix estimation under image degradation. In: *Machine Vision and Applications* 9 (1997), Nr. 5-6, 321–333. http://link.springer.com/article/10.1007/s001380050051

[TZ00] Torr, P.H.S. ; Zisserman, A.: MLESAC. In: *Comput. Vis. Image Underst.* 78 (2000), April, Nr. 1, 138–156. http://dx.doi.org/10.1006/cviu.1999.0832. – DOI 10.1006/cviu.1999.0832. – ISSN 1077–3142

[VH11] Vogiatzis, George ; Hernández, Carlos: Video-based, real-time multi-view stereo. In: *Image and Vision Computing* 29 (2011), Nr. 7, 434–441. http://www.sciencedirect.com/science/article/pii/S0262885611000138

[VL07] Valgren, Christoffer ; Lilienthal, Achim J.: SIFT, SURF and Seasons: Long-term Outdoor Localization Using Local Features. In: *EMCR*, 2007

[WB95] Welch, Greg ; Bishop, Gary: An Introduction to the Kalman Filter. Chapel Hill, NC, USA : University of North Carolina at Chapel Hill, 1995. – Forschungsbericht

[WCN+09] Williams, B. ; Cummins, M. ; Neira, J. ; Newman, P. ; Reid, I. ; Tardós, J.: A comparison of loop closing techniques in monocular SLAM. In: *Robotics and Autonomous Systems* (2009)

[Yag99] Yagi, Yasushi: Omnidirectional Sensing and Its Applications. In: *Surveys on Image Processing Technologies: Algorithms, Sensors, and Applications*, 1999 (IEICE transactions on information and systems)

[YK90] Yagi, Yasushi ; Kawato, Shinjiro: Panorama scene analysis with conic projection. In: *Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on*, IEEE, 1990, 181–187

[YKT91] Yagi, Y. ; Kawato, S. ; Tsuji, S.: Collision avoidance using omnidirectional image sensor (COPIS). In: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, IEEE, 1991, 910–915

[ZHK+07] Zhu, Jiajun ; Humphreys, Greg ; Koller, David ; Steuart, Skip ; Wang, Rui: Fast omnidirectional 3D scene acquisition with an array of stereo cameras. In: *3-D Digital Imaging and Modeling, 2007. 3DIM'07. Sixth International Conference on*, IEEE, 2007, 217–224

[ZLZH10] Zhang, Liwei ; Li, Youfu ; Zhang, Jianwei ; Hu, Ying: Homography estimation in omnidirectional vision under the L∞-norm. In: *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, IEEE, 2010, 1468–1473