Imperial College London
Department of Computing

Local Accuracy and Global Consistency for Efficient Visual SLAM

Hauke Strasdat

October 2012

Supervised by Dr. Andrew Davison

Submitted in part fulfilment of the requirements for the degree of PhD in Computing and the Diploma of Imperial College London. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

To So-Rim Lee

Abstract

This thesis is concerned with the problem of Simultaneous Localisation and Mapping (SLAM) using visual data only. Given the video stream of a moving camera, we wish to estimate the structure of the environment and the motion of the device most accurately and in real-time.

Two effective approaches were presented in the past. Filtering methods marginalise out past poses and summarise the information gained over time with a probability distribution. Keyframe methods rely on the optimisation approach of bundle adjustment, but computationally must select only a small number of past frames to process. We perform a rigorous comparison between the two approaches for visual SLAM. Especially, we show that accuracy comes from a large number of points, while the number of intermediate frames only has a minor impact. We conclude that keyframe bundle adjustment is superior to filtering due to a smaller computational cost.

Based on these experimental results, we develop an efficient framework for large-scale visual SLAM using the keyframe strategy. We demonstrate that SLAM using a single camera does not only drift in rotation and translation, but also in scale. In particular, we perform large-scale loop closure correction using a novel variant of pose-graph optimisation which also takes scale drift into account. Starting from this two stage approach which tackles local motion estimation and loop closures separately, we develop a unified framework for real-time visual SLAM. By employing a novel double window scheme, we present a constant-time approach which enables the local accuracy of bundle adjustment while ensuring global consistency. Furthermore, we suggest a new scheme for local registration using metric loop closures and present several improvements for the visual front-end of SLAM. Our contributions are evaluated exhaustively on a number of synthetic experiments and real-image data-set from single cameras and range imaging devices.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Andrew Davison for his enduring support and inspiration. He invested lots of time, provided me with helpful comments and critical remarks. I am very thankful to my colleague and unofficial second adviser José María M. Montiel from Zaragoza. I appreciate all the fruitful discussions we had. Many thanks to Kurt Konolige, whom I visited in autumn 2010 at Willow Garage, for the collaborations during the final stages of my research. I also would like to thank my previous advisors and mentors Martin Riedmiller, Sven Behnke, Cyrill Stachniss and Wolfram Burgard. They sparked my interest in computer vision/robotics and provided me with foundations essential for doctoral studies.

It was a great pleasure to do a PhD in such an inspiring and fertile environment. I wish to express my gratitude to past and current members and visitors of the Robot Vision Group. To Ankur Handa and his mathematical skills. To Steven Lovegrove; we had many chats about research, tools, programming languages and more. To Adrien Angeli who aided me with appearance-based loop closure detection. To Margarita Chli, Richard Newcombe, Renato Salas-Moreno, Gerardo Carrera, Javier Civera, Pablo Fernandez, Stefan Holzer, Klaus Strobl, Jan Jachnik, Jacek Zienkiewicz and Robert Lukierski.

I had the great opportunity to exchange ideas with various experts in the field. Thanks to Ethan Eade concerning the email correspondences about filter implementations and Lie theory. Thanks to Giorgio Grisetti and Rainer Kümmerle for the discussions about efficient optimisation. Thanks to Gabe Sibley and Christopher Mei for discussing visual SLAM and sharing details of their implementations.

I treasure the time I spend in London. My thanks go to all my friends and colleagues at Imperial College who made my studies an enjoyable experience. Cheers to the Black Lions.

I am very grateful to my family who supported me during my studies and beyond.

Contents

Chapter 1

Introduction

Imagine a digital video camera moving through the environment. At the same time it is recording a stream of images. If we make use of the rich amount of information in the images, it is possible to align the frames to each other, and in addition, to estimate the camera motion in the three-dimensional space using the underlying projective geometry. This motion path consists of a number of camera positions and orientations at different point in time; each such *pose* being associated with a camera image. Given these pose/image pairs, we can further integrate the visual measurements in order to create a consistent map of the environment. If one performs this motion estimation and mapping task concurrently, we speak of *Simultaneous Localisation and Mapping* (SLAM).

In this thesis we discuss how a camera can be used as a general purpose 3D position and mapping sensor. In particular, we tackle SLAM in *real-time*. Hence, each time a new frame arrives the camera pose as well as the map representation needs to be updated instantly. Such a real-time system enables powerful applications in the field of robotics and beyond. One such application is *augmented reality* where a live camera image is augmented with artificial content. In order to ensure that the artificial object is displayed correctly in the three dimensional space, the pose of the camera needs to be estimated which requires an accurate map. Though, the core application for real-time SLAM is mobile robotics.

©2011 NASA

(a) Mars rover

©Willow Garage

(b) Service robot

Mariordo CC BY-SA 2.0

(c) Driverless car

Figure 1.1: (a) illustrates the NASA rover 'Curiosity' which landed on Mars on 6 August 2012. Among various other sensors, it is equipped with stereoscopic cameras for obstacle avoidance and autonomous navigation. (b) shows the general-purpose service robot PR2 from Willow Garage. It consists of a mobile base and two arms for manipulation. Sensing is enabled by means of a structured-light camera. (c) shows a Toyata Prius which is equipped with a 3D laser range scanner and a set of cameras for the Google Driverless Car project.

1.1 Mobile Robotics and Real-time SLAM

Autonomous mobile robots, intelligent vehicles equipped with sensors and actuators which autonomously interact which their environment, are no longer dreams of the future. Illustrative example are the cleaning robots which have begun to enter our homes. When iRobot's first robotic vacuum cleaner 'Roomba' was introduced ten years ago, there was not much competition. Nowadays, dozens of companies have cleaning robots in their product lines and sell them at commodity prices. Apart from ethically debatable military applications, other examples of autonomous mobile robots include toy robots, space exploration rovers, mining robots and autonomous cars (see Figure 1.1). But surely, we are only at the beginning of this development; especially personal robotics is predicted to be an uprising market. While nowadays domestic robots are largely designed for special applications, some belief that general purpose service robots will have a similar impact as personal computers had in the 1980s and 1990s.

Industry robot arms in assembly lines, the older 'brothers' of mobile robots, are statically bound to a designated spot within a controlled environment and therefore operate blindly. In contrast, mobile robots do not only need a faithful representation of their own state, but also their environments. Sensing the environment is the

minimal prerequisite of even remotely intelligent mobile behaviour. Purely reactive strategies are sometimes sufficing; e.g. some robotic vacuum cleaners achieve their task without planning. They change their direction arbitrarily once an obstacles obstructs their passage and perform random walk. Though in the majority of cases, a robot needs to plan ahead in order to achieve a decent amount of intelligence and autonomy. For planning, in turn, it is required to infer and maintain an internal model of the world. Thus, autonomous mobile robots largely depend on SLAM. In most applications of mobile robotics, such as collision avoidance or path planning, it is crucial that such a model of the robot pose and its environment is not only *accurate* and *consistent*, but also *up-to-date* throughout the operation. The main challenge is to perform the required processing in a certain time frame. This time frame is determined by the desired frequency of sensor/map updates and the computational power of the available processing device. Given this limited computational budget, real-time SLAM research aims for the best possible strategies and algorithms in order to achieve the most accurate and consistent representation of the environment.

Mobile robotic tasks are often not limited to a small episodes, but the robot might operate for hours, days and beyond. Cleaning robots, for instance, typically run in a continuous operation mode. In addition, the area of operation might be very large. Hence, in order to still meet the hard real-time constraint it is important that the computational cost does not grow with the time of operation or the map size, but stays below the real-time bound. In order to emphasize this requirement, one also speaks of *constant-time SLAM*.

1.2 Vision

In this work, we focus on mapping and localisation using vision. This implies extra difficulties, but also offers several advantages.

The origin of digital image processing dates back to the mid 1960s. At that time, it was a costly and elaborate process to digitalize and process image data. With the 'Dycam Model 1', the first digital camera emerged in 1990, which could be directly connected to a personal computer.[1] In the last decade, digital cameras became widely available — mainly due to the integration in mobile phones, but

[1] It had an image resolution of 376 by 240 and was sold at a cost of approximately one thousand dollars [http://www.digicamhistory.com/1990.html].

also in terms of other consumer products such as webcams and compact digital cameras. Nowadays, digital cameras are in general inexpensive and easy to use. Opposed to other sensors, digital cameras usually have small form factors and low power consumption. Due to the lack of mechanical parts, they can be enabled to operate reliable under harsh conditions. Thus, they are suitable for an modular sensing platform. The only hardware requirement for a SLAM solution based on vision is a processing unit, a digital camera and a power supply, all which can be embedded in a small box. Instead of building a custom robotic platform with special purpose sensors, such a modular solution could for instance be temporally attached to arbitrary vehicles. To summarize, digital cameras are *small*, *inexpensive*, *low-power*, *rugged* and therefore ideal candidates for embedded applications such as general purpose navigation systems.

However, SLAM using a single camera — called *monocular camera* in order to emphasize the fact that such a device only consists of one lens and one image sensor — is difficult. In contrast to range/bearing sensors such as laser range finders, which measure distances using the *time of flight* principle[2], geometry does not pop out of the images of a monocular camera. Instead, the depth of a pixel needs to inferred from inter-frame motion. Given that a characteristic point is co-observed in two distinct camera frames, one has to estimate the unknown depth using triangulation. Furthermore, the necessity to triangulate depth over time has a significant implication for SLAM. It is impossible to measure absolute scale based on the monocular measurements only (see Figure 1.2). Finally, a camera image consists of between tens of thousands and millions of individual measurements. It is a great challenge to infer the relevant information under real-time constraints.

Despite these difficulties and in addition to the practical benefits of cameras discussed above, vision is an appealing sensing modality since it is so frequent in nature. The fact that humans and many animals rely mainly on vision for orientation and navigation tasks showed that visual SLAM is possible. Also, it allows to developed biological inspired algorithms which take nature as a role model (e.g., Milford et al., 2004). Since visual perception is a natural sense, humans have *direct access* to images. Even though a camera image consists a large amount of information, our visual cortex enables us to perceive and interpret them easily. We do not only extract geo-

[2] Sensors based on *time of flight* function as follows. They send out a light wave which is reflected by objects in the environment. Once the receiving signal is measured, the distance to the object can be estimated based on the time passed.

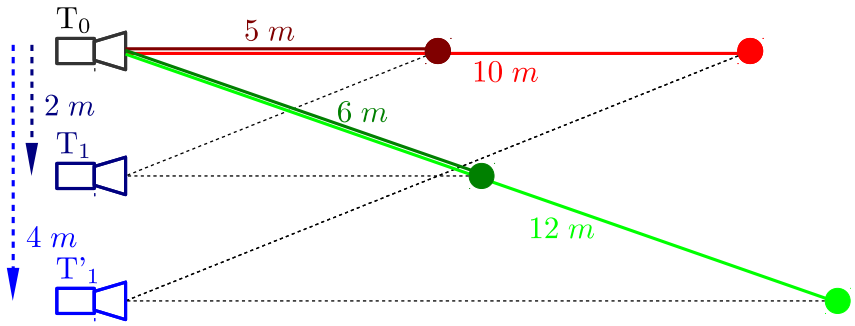


Figure 1.2: Illustration of the unknown scale factor. It is impossible to distinguish whether the camera moved 2 metres ($\mathtt{T}_0 \rightarrow \mathtt{T}_1$) and the landmarks are 5 and 6 metres away, whether the camera moved 4 metres ($\mathtt{T}_0 \rightarrow \mathtt{T}'_1$) and the landmark are 10 and 12 metres away, etc. Only the relation 2:5:6 can be recovered, while the overall *scale factor* remains unobservable.

metric informations from images, but are also able to analysis their semantic content — and so can computers. In recent years, the *computer vision* community made great progress. A vast number of systems were developed which are able to detect humans, objects, locations, events and more in images. Thus, cameras are great general purpose sensors which could enable a large amount of different applications beyond mapping.

Cameras capture an array of pixels — each representing an intensity measurements. Besides monocular cameras, there are other camera types such as light field cameras or *range imaging devices*. A range imaging device is a camera which does not only record an array of intensities, but also measure distances. Until recently, range imaging devices were still produced in low quantities and sold at corresponding prices. Available were digital *stereo cameras*, which consist of a pair of image sensors/lens assemblies and which therefore can measure depth instantly based on triangulation, and *time of flight* cameras. However, during the last two years range imaging devices had a break-through and entered the consumer mass market: End of 2010, Microsoft launched its *structured light* camera[3] called 'KINECT' and sold over eight million devices within the first two month. With the 'LG Optimus 3D',

[3]Range imaging devices using structured light embody a projector which emits a known pattern onto the environment. If a point in the pattern is detected in the camera image, the depth of the corresponding pixel can be estimated using triangulation. Structured light devices function therefore similar to stereo cameras, but have the advantage that dense point clouds can be estimated robustly even for untextured scenes.

'HTC Evo 3D', and 'Sharp Aquos SH-12C' three smartphones with integrated stereo cameras from three different manufacturers became available in 2011. Thus, algorithms which require range image measurements are now useful for a much wider community.

In this thesis, we have an emphasis on monocular SLAM; we deal with the underlying difficulties of such as depth estimation and scale drift. Here and there, however, we extend our findings to range imaging devices, in particular stereo cameras, and show how their advantages can be exploited.

1.3 A Brief Review of Visual SLAM

In the following we will briefly review the history of visual SLAM. More detailed bibliographic remarks are given at the end of the individual chapters.

In early robotics research, localisation and mapping were tackled independently. This is related to the fact they have cyclic inter-dependency. On the one hand, a map can only be created when the robot's pose is known. On the other hand, we need an accurate map representation in order to perform localisation. Nevertheless, both tasks usually need to be performed simultaneously: in this case we speak of Simultaneous Localisation and Mapping. SLAM is particularly hard since an inaccuracy in the ego motion estimate will have an negative impact on the map quality which again biases the subsequent ego motion estimate and so on. Due to inherent noise in the sensor measurements, it is clear that we cannot deal with certain entities, but we have to deal with probability distributions over the robot's pose and the world representation instead. Early attempts of SLAM were unreliable since they represented the robot's pose and the world as independent states. However, in this way one ignores the fact that the pose and the locations of the landmarks are correlated (Castellanos et al., 1997). This wrong assumption together with noisy sensor measurements leads rapidly to over-confident state estimates. The robot becomes very certain about a wrong pose estimate which will lead to fatal inconsistencies sooner or later (Davison, 1998, Sec. 5.4.4). Probably the first statistically sound formulation of SLAM was presented by Smith et al. (1987). The core principle is to represent all states, the robot's pose as well as all landmark locations, using a *joint* probability distribution. Smith et al. assumed a uni-modal, in particular a multi-

variate Gaussian distribution, and suggested to perform state estimation using the *Extended Kalman Filter* (EKF). In the 1990s, the EKF formulation emerged as the standard approach for SLAM and stood the test in real robot applications (Leonard & Durrant-Whyte, 1991; Betgé-Brezetz et al., 1996; Castellanos, 1998; Davison, 1998; Newman, 1999). The limitations of EKF-SLAM, in particular the quadratic time and space complexity of the algorithm with respect to the number of landmarks in the map, as well as the occurrence of inconsistencies due to the linearisation of non-linear sensor and motion models, were well-studied in the past and many improved algorithms were presented subsequently (Julier & Uhlmann, 2001; Thrun et al., 2002; Paskin, 2003; Montemerlo & Thrun, 2003)

Various sensors were used for SLAM including sonar, laser range finders, and multi-sensor approaches. Pioneer work on robotic navigation using *vision* date back to the 1970's. Moravec's robotic cart (1980) was equipped with a camera on a slider for depth estimation and performed autonomous navigation and obstacle avoidance. Many SLAM frameworks using cameras were presented in the past 15 years; we will give a few examples. Davison (1998) employed a mobile robot with a active stereo head and performed navigation and SLAM. In particular, he followed an active vision approach where selective visual features are fixated during navigation. Se et al. (2002) also used a stereo-camera on a mobile robot, but performed bottom-up feature tracking using SIFT (Lowe, 1999). They initialized the ego motion using wheel odometry but refined it by means of least-square minimisation. Both approaches integrated visual information with wheel odometry in order to build two-dimensional maps. Milford et al. (2004; 2008) presented RatSLAM, a biological-inspired visual SLAM approach which maintains a two-dimensional *topological* map. In their earlier approach, visual data was fused with sonar and odometry measurements. In contrast, Milford & Wyeth (2008) only relied on data from a single camera which was mounted on an automobile.

In this work we regard *visual SLAM* in the most general sense. Opposed to the classic SLAM approach in robotics where one tracks the pose of a robot on the ground plane, we aim to estimate the camera pose which moves freely in the three dimensional space. Instead of creating two-dimensional map by either representing a slice of the world or projecting 3D landmarks onto the ground, we create a full three dimensional representation of the environment. Furthermore, we assume that there is *no* odometry sensoring available which allow us to calculate an accurate

motion prediction and therefore we have to purely rely on the information from the image measurements.

In an early approach, Harris & Pike (1987) were able to estimate the ego motion of a single camera and create a three dimensional map in real-time. They performed iterative inference using Kalman filtering, but assumed independent states and thus did not model the correlation between the features and the camera pose. Almost ten years ago, Chiuso et al. (2002) as well as Davison (2003) presented real-time frameworks using a monocular camera only; they estimated structure and motion jointly using an EKF. While Chiuso et al. relied on the assumption that a scene of 20-40 predefined features is in view all the time, Davison's *MonoSLAM* is a fully automated framework for visual SLAM including feature tracking using correlation-based patch matching, initialisation of new features, estimation of the unknown depth of newly initialised features, redetection of temporally occluded features and small scale loop closures. This approached allowed to build a map of dozens of features in real-time while performing a browsing motion in a local workspace.

Visual SLAM research is closely related to scientific discipline of *photogrammetry* and *structure from motion* research of the computer vision community. In photogrammetry, one tries to extract geometric information out of photographs. From the 1950s, a core area of application was the interpretation and evaluation of aerial photographs in the context of cartography. Nowadays, an illustrative example is Google Maps where a symbolic map of road networks and labels can be overlayed with photographs taken from aircrafts and satellites. In order to create such an overlay, one need to align the photographs with the symbolic map. In the most simplistic setting, i.e. under a flat-earth assumption and knowing the configuration of the camera's lens/sensor assembly as well as the height and angle from which the photograph was taken, one can calculate the actual distance of any two points visible in the image by the intercept theorem. More sophisticated methods employ several overlapping photographs and are able to reconstruct the three dimensional structure of the Earth's surface. Given a set of corresponding points among the images, a three dimensional point cloud is created. The central technique is *bundle adjustment* (Brown, 1958; Triggs et al., 1999). It is an iterative optimisation technique which aims to minimize the distance between reprojections of the three dimensional model and the associated points in the image. Photogrammetry is largely overlapping with structure from motion research which emerged in the 1980s within the

field of computer vision. The core difference is that computer vision research operates from an artificial intelligence perspective and thus tries to eliminate the human in the loop. An illustrative example is Agarwal et al.'s 'Rome in a day' (2009). It is a fully automated framework which queries thousands images of a given location from the internet, finds correspondences among them, estimates their relative configuration and obtains a three dimensional model of hundreds thousands of point — using a large scale optimisation based on bundle adjustment.

Thus, visual SLAM using recursive Kalman filtering and structure from motion using bundle adjustment are largely equivalent. In a nutshell, both approaches minimise the same cost function, the sum of squares of reprojection errors. Both approaches estimate motion and structure in the full three dimensional space and do not incorporate any additional priors besides the image data. The core difference lies within the problem formulation. SLAM is usually perceived as an *online* method. Representative SLAM applications such as autonomous navigation or augmented reality require pose and map estimates which are up-to-date all the time. Frames arrive consecutively, and once a new frame arrives, the joint state must be updated instantly. In contrast, structure from motion is usually a *batch approach*. First, all data is collected from a set of images. Afterwards, a three dimensional representation is estimated using an extensive offline optimisation. In online SLAM methods such as filtering, the emphasis is to estimate a probability distribution over the current pose and the map which is *statistically valid*, e.g. not overconfident. On the other hand, batch approaches such as bundle adjustment solve the problem from scratch and hence do not need to directly deal with probability distributions. Here, the main focus is *accuracy*.

The close relation between offline structure from motion techniques such as bundle adjustment and online iterative state estimation in SLAM was discovered and exploited by several researches from various perspectives. Examples include the variable state dimension filter approach of McLauchlan & Murray (1995), the related sliding window filter of Sibley et al. (2008), the exactly sparse extended information filter of Walter et al. (2007), smoothing and mapping by Dellaert & Kaess (2006) and the corresponding incremental approaches by Kaess et al. (2008, 2012). In *visual odometry* approaches such as Nistér et al. (2004), bundle adjustment is applied in a sliding window. This way an accurate incremental motion estimate can be calculated in real-time; but global consistency is not ensured. At the other end of

the spectrum, there are *pose-graph optimisation* approaches such as Agrawal (2006) and Grisetti et al. (2007) which originates from Lu & Milios (1997). Pose-graph optimisation is an efficient batch method for loop closure correction where feature measurements are eliminated and approximated by relative pose-pose constraints.

In particular, however, Klein & Murray (2007) made a large contribution to the mayor amalgamation of both fields. In their stand-out work *Parallel Tracking And Mapping* (PTAM), they separated the visual SLAM problem into two subtasks using a multi-threading approach. In one thread, they track the motion of a single camera given a three dimensional model of the world. Since this pose tracking given a known model involves an relatively inexpensive optimisation, it can be performed in real-time on every single frame using a scene model consisting of hundreds of points. A key aspect of this system is that a number of *keyframes* are extracted out of the image stream. These keyframes are carefully selected frames which cover the area of operation. In a second thread, PTAM performs joint optimisation over all points in the map and a number of keyframes using bundle adjustment. Importantly, this optimisation need not to performed at frame rate but at lower frequency. The main limitation of PTAM is that it is only suitable for a small area of operation — in order to restrict the number of keyframes and therefore the computational requirement of the optimisation back-end.

PTAM surpassed the MonoSLAM approach in terms of accuracy and robustness; but several enhancements of filter-based visual SLAM were present in subsequent years such as Montiel et al. (2006), Davison et al. (2007), Eade & Drummond (2007) and Pietzsch (2008). Meanwhile, in stereo vision-based SLAM robust and accurate approaches were presented which enable large scale mapping and rely on the keyframes and batch optimisation such as Konolige & Agrawal (2008) and Lim et al. (2011). To recapitulate, the most accurate solution of structure and motion estimation is clearly the joint optimisation of all available information in a batch approach. In order to achieve real-time performance, however, both SLAM approaches, filtering and keyframe bundle adjustment, sparsify the problem in different ways. While filtering includes every single frame in the estimation, keyframes methods only optimises over a selected number of frames which then allows to increase the number of points in the three dimensional model by a magnitude. So the question remained whether iterative filtering or keyframe bundle adjustment is the method of choice for real-time visual SLAM.

1.4 Efficiency, Accuracy and Consistency

One often distinguishes between two phases of SLAM: The *exploration* phases where the robot is travelling through unknown parts of the environment and the *loop closure*, hence the event when the robot is returning after some exploration to a known location. Loop closure is commonly perceived as the hard problem of SLAM; it is the typically test case when evaluating particular SLAM approaches. Due to the inherent noise in sensor measurements, the pose estimate is prone to drift during exploration so that there will be a significant difference between the pose estimate and the true pose after some time. Once the robot/camera returns to a previous visited location, it is the challenge to ensure a consistent map representation in spite of the drift. Filter-based SLAM approaches tackle this problem by modelling the predicted drift using a joint probability distribution. While the uncertainty of this distribution grows during exploration, it is likely to peak when a known place is revisited. Such a metric approach can handle small to middle scale loop closures well. However, if the robot is only travelling long enough, its pose estimate becomes so uncertain that it cannot be modelled accurately and at the same time efficiently any more.[4] Instead, large-scale loop closures are very effectively detected using appearance information (Nister & Stewenius, 2006; Angeli et al., 2008; Cummins & Newman, 2009). Once the robot enters a scene which appears similar to a place it visited before, a loop closure hypothesis gets triggered. After a positive verification, the drift over the chain of motion is corrected in order to achieve a consistent map representation (Lu & Milios, 1997; Grisetti et al., 2007; Konolige & Agrawal, 2008). Therefore, we will distinguish between small scale loop closures which can be detected using metric information and large scale loop closures which rely on appearance-based place recognition.

Even though drift during exploration is unavoidable, it is beneficial to keep it as small as possible. Therefore, we will analyse the *building blocks* of visual SLAM: the joint motion and structure estimation over a short distance. Our question of interest is how the incremental motion can be estimated most accurately given a limited computational budget.

[4] On the one hand, EKF-based approaches and variants have the problem that the pose distribution will become highly non Gaussian after some time due to the non-linear sensor measurements and camera motion. On the other hand, non-parametric approaches using particle filters (Montemerlo & Thrun, 2003; Sim et al., 2005; Eade & Drummond, 2006) will fail eventually too since the required number of particles will exceed all bounds.

Despite the exploration phase and loop closure phase, a different type of motion pattern is prevalent: a local motion where the camera/robot is repeatedly browsing over a restricted area (which is especially common for augmented reality applications). One particular challenge is to avoid that the map grows constantly. This can be achieved using top-down feature matching (Davison, 2005; Klein & Murray, 2007). One actively searches for previous initialized map points and only adds new features when necessary. A related phase is the one *after* a large scale loop closure. When a known place is revisited, one needs to register temporally distinct, but spatially overlapping scene reconstructions. This is commonly achieved by a combination of exploration and appearance-based loop closures (Konolige & Agrawal, 2008; Mei et al., 2009; Lim et al., 2011) Two separate map segments are maintained which are linked with a number of constraints. However, we believe local registration should be treated in a unified way regardless of whether the camera performs local browsing or revisits a know place after a long loop. Thus, we will discuss how to join the temporally distant, but spatially overlapping map areas and therefore exploit the available geometric information in the previous constructed map segment.

In a nutshell, previous approaches concentrated on different aspects of visual SLAM. MonoSLAM (Davison, 2003; Davison et al., 2007) and PTAM (Klein & Murray, 2007) deal well with local browsing motion, but have difficulties with rapid exploration and are restricted to limited workspaces. Visual odometry frameworks (Nistér et al., 2004; Konolige et al., 2007) were designed for incremental motion estimation, but do not enforce global consistency. Some approaches combined visual odometry with pose-graph optimisation (e.g. Steder et al., 2007; Konolige & Agrawal, 2008) so that they scale well for large scale mapping and loop closing. However, those approaches are not suitable for local browsing motion since they do not reuse previous reconstructed geometry. Instead, we aim for a unified framework for efficient visual SLAM which can deal with all different kinds of motion pattern. Thereby, we mainly concentrate on the optimisation back-end. How can we best perform structure and motion estimation in order to achieve a local accurate and global consistent map under strict time constraints?

1.5 Contributions

In this thesis, we present a rigorous comparison of recursive Gaussian filtering versus keyframe bundle adjustment. Thereby, we concentrate on the local building block of SLAM — the joint estimation of structure and motion over a short distance. In particular we show that the main accuracy comes from a large number of points; the number of intermediate frames only has a minor impact. We conclude that keyframe bundle adjustment is superior to filtering. While filtering can approach the accuracy of bundle adjustment, the predominance of keyframe bundle adjustment is mainly a cost argument. A large set of simulation experiments were performed. The experiments consider different scene/motion settings, a monocular as well as a stereo camera model and are therefore widely applicable.

On the basis of these experimental results, we develop an efficient framework for monocular SLAM using the keyframe strategy. We integrate keyframe bundle adjustment with a novel approach for pose-graph optimisation. In particular, we show that monocular SLAM does not only drift in rotation and translation, but also in scale. Consequently, we employ a loop closure scheme which also takes scale drift into account. Starting from this two stage approach, where we tackle local motion estimation and loop closing alternately, we develop a unified framework for efficient visual SLAM. We use a novel *double window* scheme and solve bundle adjustment and pose-graph optimisation jointly by minimising a common cost term. This constant-time approach is suitable for general motion pattern such as local browsing, exploration and long loops. It achieves the local accuracy of bundle adjustment while maintaining global consistency. Furthermore, we present a new strategy for local registration using metric loop closures. The framework is tested exhaustively on a number of synthetic experiments and real-image data-sets from monocular cameras, stereo cameras as well as structured light devices.

In addition, we suggest several improvements for the visual front-end. We show how feature-based visual SLAM is enhanced using accurate dense tracking methods which can be computed very efficiently on modern GPUs. In particular, we employ variational optical flow for monocular SLAM and Lucas-Kanade tracking on a three dimensional point cloud for SLAM using range imaging devices and integrate both with sparse feature tracking. Also, we organise visual measurements in a quadtree and present a new traversal method which ensures uniform feature selection.

1.6 Publications

The core of the thesis relies on the following peer-reviewed publications:

- H. Strasdat, J. M. M. Montiel & A. J. Davison: **Monocular SLAM: Why Filter?** In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2010. ***(ICRA best vision paper)***
- H. Strasdat, J. M. M. Montiel & A. J. Davison: **Scale-drift Aware Large Scale Monocular SLAM.** In *Proceedings of Robotics: Science and Sytems (RSS)*, June 2010, and in *Y. Matsuoka, H. Durrant-Whyte, J. Neira (editors)*, MIT Press, September 2011.
- H. Strasdat, A. J. Davison, J. M. M. Montiel & K. Konolige: **Double Window Optimisation for Constant Time Visual SLAM.** In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, November 2011.
- H. Strasdat, J. M. M. Montiel & A. J. Davison: **Visual SLAM: Why Filter?** In *J.-M. Frahm, M. Pantic (editors): Image and Vision Computing*, Volume 30, Issue 2, February 2012.

1.7 Structure

In the subsequent chapter, we will give an overview of theoretical preliminaries which are relevant for the remainder of the thesis. In particular, we present least-squares optimisation, probabilistic state estimation and recursive filtering. Furthermore, we introduce Lie groups as a generalisation of Euclidean vector spaces and elaborate how optimisation can be performed on them. In Chapter 3, we tackle monocular exploration. This offers a smooth introduction to the problem of visual SLAM and bundle adjustment; it allow us to introduce concepts such as camera models, and the quad tree based feature selection which are required subsequently. Afterwards, we present the rigorous comparison of filtering versus keyframe bundle adjustment in Chapter 4. In Chapter 5, we complete the framework for monocular exploration to a large SLAM framework by tackling the problems of large scale loop closures and scale drift. Then, in Chapter 6, we will present a novel, uniform and scalable

approach for visual SLAM using double window optimisation. Finally, in Chapter 7 we discuss our outcomes and future work.

Chapter 2

Preliminaries

In which we first revise a number of basic mathematical concepts and second introduce a generalisation over the Euclidean space: Lie groups.

In an attempt to make this document as self-contained as possible, and also to clarify notation, a number of mathematical concepts shall be presented including multivariate differentiation, Taylor series, numerical optimization techniques, probabilistic state estimation and its relation to least squares problems. Motivated as a generalisation over the Euclidean vectors space, we introduce Lie groups and their underlying related concepts.

2.1 Some Revision of Calculus

2.1.1 Multivariate Differentiation

A function $F : \mathbb{R}^n \to \mathbb{R}$, which maps a vector onto a scalar, is called a *scalar field*. Furthermore, a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, which maps a vector onto a vector, is called a *vector field*.

The first derivative of a scalar field $F : \mathbb{R}^n \to \mathbb{R}$ is a vector field $\nabla F : \mathbb{R}^n \to \mathbb{R}^n$.

It is called the *gradient* of F and defined as

$$\nabla F(\mathbf{x}) := \left(\frac{\partial F(\mathbf{x})}{\partial x_1}, ..., \frac{\partial F(\mathbf{x})}{\partial x_n}\right)^\top. \tag{2.1}$$

The second derivative of F is a function $\mathtt{H}_F : \mathbb{R}^n \to \mathbb{R}^{(n\times n)}$ which maps an n-vector onto a $n \times n$ matrix. It is the *Hessian* of F:

$$\mathtt{H}_F(\mathbf{x}) := \begin{bmatrix} \frac{\partial^2 F(\mathbf{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial x_1 x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F(\mathbf{x})}{\partial x_1 x_n} & \cdots & \frac{\partial^2 F(\mathbf{x})}{\partial x_n^2} \end{bmatrix}. \tag{2.2}$$

Furthermore, the first derivative of a vector field $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is the *Jacobian* $\mathtt{J}_\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^{(m\times n)}$ which is defined as:

$$\mathtt{J}_\mathbf{f}(\mathbf{x}) := \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{bmatrix}. \tag{2.3}$$

Thus, the transpose of the gradient vector can be seen as special case of the Jacobian matrix (with $m = 1$). The second derivative of a vector field $\mathbf{f}$ is a function $\mathtt{H}_\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^{(n\times m\times n)}$ which maps a vector onto a three dimensional array or a third-order *tensor*. Again, $\mathtt{H}_\mathbf{f}$ is called the Hessian (tensor) of $\mathbf{f}$.

2.1.2 Taylor Series

Let $f : \mathbb{R} \to \mathbb{R}$ be an infinitely differentiable function. The power series

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(x)}{2!}(x-a)^2 + ... = \sum_{k=0}^{\infty} \frac{f^{(n)}(a)}{k!}(x-a)^k \tag{2.4}$$

is called the *Taylor series* of f at a. If f is analytic[1], then its Taylor series has a positive radius of convergence r and if furthermore $|x-a| < r$, it holds that

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(n)}(a)}{k!}(x-a)^k. \tag{2.5}$$

In practice, we often approximate a function f in the neighbourhood of a point a using a finite series, the nth-order Taylor expansion:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + ... + \frac{f^{(n)}(a)}{n!}(x-a)^n = \sum_{k=0}^{n} \frac{f^{(n)}(a)}{k!}(x-a)^k. \tag{2.6}$$

[1] A function $f : \mathbb{R} \to \mathbb{R}$ is analytic in x_0
$:\Leftrightarrow \exists\, a_0, ..., a_k, ... \in \mathbb{R}\ \forall\, x$ in the neighbourhood of $x_0 : f(x) = \sum_{k=0}^{\infty} a_k (x - x_0)^k$.

There is a generalised version of Taylor series for scalar fields F. For instance, the second-order Taylor expansion of F around $\mathbf{a}$ would be

$$F(\mathbf{a}) + (\mathbf{x}-\mathbf{a})^\top \nabla F(\mathbf{a}) + \frac{1}{2}(\mathbf{x}-\mathbf{a})^\top \mathtt{H}_F(\mathbf{a})(\mathbf{x}-\mathbf{a}). \tag{2.7}$$

2.2 Introduction to Optimisation

Let $F : \mathbb{R}^n \to \mathbb{R}$ be a scalar field. In typical optimisation problems, we would like to find the minimum of F:

$$\min_{\mathbf{x}\in\mathbb{R}^n} F(\mathbf{x}). \tag{2.8}$$

For a general scalar field, even if we assume it is infinitely differentiable, we are not guaranteed to find such a global minimum in countable many steps. Therefore, one often focuses on finding local minima in the neighbourhood of an initial guess $\mathbf{x}_0$ instead. If $\langle \bar{\mathbf{x}}, F(\bar{\mathbf{x}})\rangle$ is a local minimum of F, then $\nabla F(\bar{\mathbf{x}}) = 0$, which is called the *necessary condition*. Furthermore, if $\nabla F(\bar{\mathbf{x}}) = \mathbf{0}$ and $\mathtt{H}_F(\bar{\mathbf{x}})$ is *positive definite*, thus $\forall_{\mathbf{y}\in\mathbb{R}^n\setminus\{\mathbf{0}\}} \mathbf{y}^\top \cdot \mathtt{H}_F(\bar{\mathbf{x}}) \cdot \mathbf{y} > 0$, then $\langle \bar{\mathbf{x}}, F(\bar{\mathbf{x}})\rangle$ is a local minimum. This is the sufficient condition.

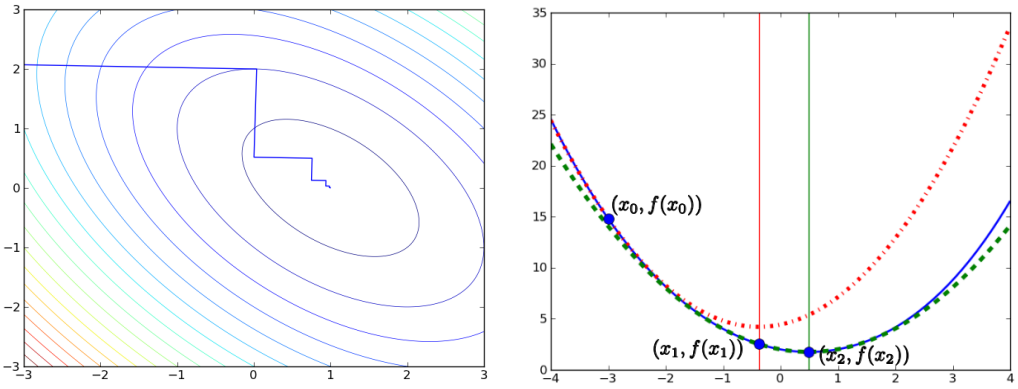
2.2.1 Gradient Descent

Let us assume that F is differentiable and we would like to find a local minimum of F in the neighbourhood of $\mathbf{x}^{(0)}$. In the method of *gradient descent*, we walk iteratively, $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}, ...$, along the direction of the negative gradient $-\nabla F(\mathbf{x}^{(k)})$. Thus, we employ the following update rule:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla F(\mathbf{x}^{(k)}). \tag{2.9}$$

Typically, the factor $\alpha_k > 0$ is selected in a way such that $F(\mathbf{x}^{(k+1)}) \ll F(\mathbf{x}^{(k)})$.[2] If no such α_k exists, the minimum is reached. For instance, we could select α_k using a *back-tracking line search*: We initialise $\alpha_k = 1$ and then iteratively downsize it, e.g. $\alpha_k \leftarrow \frac{1}{2}\alpha_k$, until the condition $F(\mathbf{x}^{(k+1)}) \ll F(\mathbf{x}^{(k)})$ is fulfilled or gradient descent is converged ($\alpha_k < \epsilon$). Gradient descent with line search is guaranteed to converge locally. However, the convergence rate can be low, especially close to the minimum (see Figure 2.1(a)).

[2] Here, $\ll$ means sufficiently small. In theory, the condition $F(\mathbf{x}^{(k+1)}) < F(\mathbf{x}^{(k)})$ is not sufficient to guarantee convergence. We won't elaborate here but refer the reader to the Wolfe conditions (No-



(a) Gradient descent (b) Newton method

Figure 2.1: (a) Method of gradient descent illustrated on a quadratic form (as defined below in equation (2.14)). The method always walks along the direction of steepest descent, which leads to a "zig-zagging" and thus potential slow convergence close to the minimum. Here, a near optimal line search is used. (b) The Newton method is illustrated on a one-dimensional higher-order polynomial (solid blue curve). The neighbourhood around the initial guess x_0 is approximated with a one-dimensional positive definite quadratic form: a parabola (red dashed curve). The initial update $x_0 \to x_1$ is performed by stepping to the minimum of the parabola (vertical red line). Also, a second update $x_1 \to x_2$ is shown (green parabola, green vertical line) which brings the estimate very close to the optimum already.

2.2.2 Newton Method

A more efficient approach is the Newton method which requires that F is twice differentiable. This method cannot distinguish between minima, saddle points and maxima. In the following, however, we will assume that our initial estimate $\mathbf{x}_0$ is in the neighbourhood of a local minimum at $\bar{\mathbf{x}}$. In other word, we require that the Hessian $\mathtt{H}_F$ is positive semi-definite in this neighbourhood. Due to the necessary condition, $\bar{\mathbf{x}}$ is a root of the gradient ∇F (i.e. a vector $\bar{\mathbf{x}}$ such that $\nabla F(\bar{\mathbf{x}}) = \mathtt{0}$). Since we assume that $\mathbf{x}_0$ is in the neighbourhood of $\bar{\mathbf{x}}$, we can approximate $\nabla F(\bar{\mathbf{x}})$ using the first-order Taylor expansion,

$$\nabla F(\bar{\mathbf{x}}) \approx \nabla F(\mathbf{x}_0) + \mathtt{H}_F(\mathbf{x}_0)(\bar{\mathbf{x}} - \mathbf{x}_0) \ . \tag{2.10}$$

Because $\nabla F(\bar{\mathbf{x}}) = \mathbf{0}$, we get $\bar{\mathbf{x}} \approx \mathbf{x}_0 - \mathtt{H}_F^{-1}(\mathbf{x}_0)\nabla F(\mathbf{x}_0)$. This leads to the recursive update formula:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathtt{H}_F^{-1}(\mathbf{x}^{(k)})\nabla F(\mathbf{x}^{(k)}) \ . \tag{2.11}$$

cedal & Wright, 2006).

Defining the incremental update as $\boldsymbol{\delta} := \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$, we can perform the *Newton method* by repetitively solving the following linear system

$$\mathtt{H}_F(\mathbf{x}^{(k)})\boldsymbol{\delta} = -\nabla F(\mathbf{x}^{(k)}) \tag{2.12}$$

followed by an additive update

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta} \ . \tag{2.13}$$

To get a second view on the method, let us consider the *quadratic form*, a vector field generalization of the quadratic function:

$$\frac{1}{2}\mathbf{x}^\top \mathtt{A}\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c \ . \tag{2.14}$$

If $\mathtt{A}$ is symmetric and positive semi-definite, the quadratic form is minimal for $\mathtt{A}\mathbf{x} = \mathbf{b}$ (Shewchuk, 1994). Looking at equation (2.12), it becomes clear that the Newton method approximates the function F at $\mathbf{x}^{(k)}$ with a quadratic form with $\mathtt{A} = \mathtt{H}_F(\mathbf{x}^{(k)})$ and $\mathbf{b} = -\nabla F(\mathbf{x}^{(k)})$. Hence, if F happens to be a quadratic form, the Newton method will converge in one iteration. Note that any function is approximately quadratic around its minimum (= "bowl-shaped") if it is twice differentiable. Thus in contrast to gradient descent, the Newton method converges especially fast in the neighbourhood of the minimum. Figure 2.1(b) illustrates this using a one dimensional function.

2.2.3 Gauss-Newton Method

For high-dimensional problems, it is often intractable to calculate the Hessian $H_F(\mathbf{a})$. We now consider an efficient variant of the Newton method which requires that the scalar field $F : \mathbb{R}^n \to \mathbb{R}$ is of the following class:

$$F(\mathbf{x}) = a \cdot \mathbf{d}(\mathbf{x})^\top \mathtt{\Lambda}\mathbf{d}(\mathbf{x}) \ , \tag{2.15}$$

with $a > 0$, $\mathbf{d} : \mathbb{R}^n \to \mathbb{R}^m$ being a twice differentiable vector field, and $\mathtt{\Lambda} \in \mathbb{R}^{m \times m}$ being a symmetric, positive semi-definite matrix. We are interested in the minimum of F. Even though this optimization domain seem to be very special, it covers a large problem class. Especially, it covers least square problems where we would like to estimate a model parameters $\mathbf{x}$ by minimising a quadratic cost $\sum_i (\mathbf{z}_i - \hat{\mathbf{z}}_i(\mathbf{x}))^2$ as we will see later.

Without loss of generality, we can assume that $a = \frac{1}{2}$ since scaling F does not change the position of its minima. Due to the product rule, the first derivative of F becomes:

$$\nabla F = \frac{1}{2}(\mathbf{d}(\mathbf{x})^\top \Lambda \mathtt{J}_\mathbf{d}(\mathbf{x}))^\top + \frac{1}{2}(\mathtt{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathbf{d}(\mathbf{x})) = \mathtt{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathbf{d}(\mathbf{x}) \ , \tag{2.16}$$

using the fact that Λ is symmetric. Again by means of the product rule, the second derivation of F is

$$\mathtt{H}_F(\mathbf{x}) = \mathtt{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathtt{J}_\mathbf{d}(\mathbf{x})) + \mathtt{H}_\mathbf{d} \Lambda \mathbf{d}(\mathbf{x}) \ , \tag{2.17}$$

with $\mathtt{H}_\mathbf{d}$ being the Hessian tensor of $\mathbf{d}$. The *Gauss-Newton* method approximates the Hessian of F as

$$\mathtt{H}_F(\mathbf{x}) \approx \mathtt{J}_\mathbf{d}(\mathbf{x})^\top \Lambda \mathtt{J}_\mathbf{d}(\mathbf{x})) \ . \tag{2.18}$$

This approximation behaves well when $\mathbf{d}(\mathbf{x})$ is small, since then the second term of equation (2.17) is negligible. It is especially crucial that this property holds true around the minimum of F where the Newton method approaches quadratic convergence. To summarize, in Gauss-Newton the linear system of equation (2.12) is approximated by the *normal equation*

$$(\mathtt{J}_\mathbf{d}^\top \Lambda \mathtt{J}_\mathbf{d})\boldsymbol{\delta} = -\mathtt{J}_\mathbf{d}^\top \Lambda \mathbf{d} \ . \tag{2.19}$$

2.2.4 Levenberg-Marquardt

Let us recapitulate that Newton-type methods works well close to the minimum, but elsewhere they might be attracted by local maxima and saddle points too. On the contrary, gradient descent converges globally but performs especially poor close to the minimum. This leads to the Levenberg-Marquardt algorithm which interpolates between Gauss-Newton and gradient descent by altering the normal equations as follows:

$$\left(\mathtt{J}_\mathbf{d}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{d} + \mu \mathtt{I}\right) \boldsymbol{\delta} = -\mathtt{J}_\mathbf{d}^\top \Lambda_\mathbf{z} \mathbf{d} \ . \tag{2.20}$$

The parameter $\mu > 0$ rotates the update vector $\boldsymbol{\delta}$ towards the direction of the steepest descent. If μ approaches zero, Levenberg-Marquardt approaches pure Gauss-Newton. On the other hand, if μ approaches infinity, the matrix $\left(\mathtt{J}_\mathbf{d}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{d} + \mu \mathtt{I}\right)$ approaches a diagonal matrix with infinite trace. Thus, for $\mu \to \infty$, Levenberg-Marquardt approaches an gradient descent update

$$\boldsymbol{\delta} = -\alpha \mathtt{J}_\mathbf{d}^\top \Lambda_\mathbf{z} \mathbf{d} \ , \tag{2.21}$$

with a minimal step-size $\alpha \to 0$, which is bound to reduce the error by an infinitesimal small step if the minimum is not reached yet.

The Levenberg-Marquardt algorithm is performed as follows: Only if the update $\mathbf{x}^{(k)} + \boldsymbol{\delta}$ reduces the error, i.e. $F(\mathbf{x}^{(k)} + \boldsymbol{\delta}) \ll F(\mathbf{x}^{(k)})$, we accept the update $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}$. In addition, Levenberg-Marquardt assumes we are approaching the local minimum and hence μ is reduced to strengthen the influence of Gauss-Newton. However, if the update $\mathbf{x}^{(k)} + \boldsymbol{\delta}$ does not reduce the error, it is rejected, and we try again with a larger μ, i.e. with smaller step size and an update direction more towards the steepest descent direction.

2.3 Probabilistic State Estimation and Filtering

Assume that we would like to estimate a parameter vector $\mathbf{x}$ given a measurement vector $\mathbf{z}$. Furthermore, we know the form of the likelihood function $p(\mathbf{z}|\mathbf{x})$. This function quantifies the probability that we make a particular measurement $\mathbf{z}$ given that the parameter is $\mathbf{x}$. This state estimation problem can be visualized using a simple graphical model (see Figure 2.2). The most probable solution is the set of values $\mathbf{x}$ which maximises this likelihood, which is equivalent to minimising the negative log-likelihood:

$$\arg\max_{\mathbf{x}} p(\mathbf{z}|\mathbf{x}) = \arg\max_{\mathbf{x}} \log p(\mathbf{z}|\mathbf{x}) = \arg\min_{\mathbf{x}}(-\log p(\mathbf{z}|\mathbf{x})) \ . \tag{2.22}$$

In many optimisation problems, negative log-likelihood is known as *energy* and the goal is to minimise it, which is fully equivalent to maximising the probability of the solution – under the assumption of a uniform prior on $\mathbf{x}$. In the common case that the likelihood function is a product of several *factors*, $p(\mathbf{z}|\mathbf{x}) = \prod_{k=1}^{K} \phi_k(\mathbf{x}, \mathbf{z})$, the energy is a sum of negative log-factors:

$$\arg\max_{\mathbf{x}} \left(\prod_{k=1}^{K} \phi_k(\mathbf{x}, \mathbf{z}) \right) = \arg\min_{\mathbf{x}} \left(-\sum_{k=1}^{K} \log \phi_k(\mathbf{x}, \mathbf{z}) \right) \ . \tag{2.23}$$

2.3.1 State Estimation using Gauss-Newton

We now speak in more specific but still very widely applicable terms, and assume that the likelihood distribution $p(\mathbf{z}|\mathbf{x})$ is jointly Gaussian, and thus has the distribution

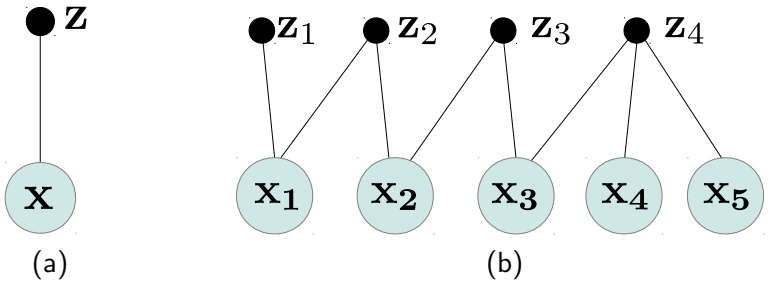


Figure 2.2: (a) Illustration of a general state estimation problem as graphical model. Observable variables, here the measurement $\mathbf{z}$, are visualised as small black circles, while the parameters to estimate, here $\mathbf{x}$, are within blue circles. (b) *Factor graph* representation of an example least squares problem. The energy consists of four measurement functions: $\hat{\mathbf{z}}_1(\mathbf{x}_1)$,$\hat{\mathbf{z}}_2(\mathbf{x}_1, \mathbf{x}_2)$,$\hat{\mathbf{z}}_3(\mathbf{x}_2, \mathbf{x}_3)$ and $\hat{\mathbf{z}}_4(\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5)$. This is reflected by four constraints in the graph: A unary constraint on $\mathbf{x}_1$, two binary constraints, and a ternary one between $\mathbf{x}_3$, $\mathbf{x}_4$ and $\mathbf{x}_5$.

(up to a constant of proportionality):

$$p(\mathbf{z}|\mathbf{x}) \propto \exp(-(\mathbf{z} - \hat{\mathbf{z}}(\mathbf{x}))^\top \Lambda_\mathbf{z} (\mathbf{z} - \hat{\mathbf{z}}(\mathbf{x}))) \ , \tag{2.24}$$

where $\Lambda_\mathbf{z} = \Sigma_\mathbf{z}^{-1}$, the *information matrix* or inverse of the *measurement covariance matrix* $\Sigma_\mathbf{z}$ of the likelihood distribution, and $\hat{\mathbf{z}}(\mathbf{x})$ is the 'measurement function' or 'forward model' which computes (predicts) the distribution of measurements $\mathbf{z}$ given a set of parameters $\mathbf{x}$. The negative log-likelihood, or energy, $-\log p(\mathbf{z}|\mathbf{x}) := \chi^2(\mathbf{x})$ therefore is the following quadratic expression:

$$\chi^2(\mathbf{x}) = (\mathbf{z} - \hat{\mathbf{z}}(\mathbf{x}))^\top \Lambda_\mathbf{z} (\mathbf{z} - \hat{\mathbf{z}}(\mathbf{x})) \ . \tag{2.25}$$

Note that it matches the right hand side of equation (2.15) with $\mathbf{d} := \mathbf{z} - \hat{\mathbf{z}}$ approximating zero at the minimum. Therefore, it is suitable for Gauss-Newton-type optimisation methods such as Levenberg-Marquardt.

If the likelihood is a product of several factors ϕ_k and the energy is a sum of negative log-terms (equation (2.23)), the measurement information matrix $\Lambda_\mathbf{z}$ is block-diagonal,

$$\Lambda_\mathbf{z} = \begin{bmatrix} \Lambda_{\mathbf{z}_1} & 0 & \dots & 0 \\ 0 & \Lambda_{\mathbf{z}_2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \Lambda_{\mathbf{z}_K} \end{bmatrix} =: \operatorname{diag}(\Lambda_{\mathbf{z}_1}, ..., \Lambda_{\mathbf{z}_2}) \quad \text{with} \quad \mathbf{z} = \begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_K \end{pmatrix} \ , \tag{2.26}$$

and the Gaussian energy $\chi^2(\mathbf{x})$ simplifies to:

$$\chi^2(\mathbf{x}) = \sum_{k=1}^{K} (\mathbf{z}_k - \hat{\mathbf{z}}_k(\mathbf{x}))^\top \Lambda_{\mathbf{z}_k} (\mathbf{z}_k - \hat{\mathbf{z}}_k(\mathbf{x})) \ . \tag{2.27}$$

This kind of minimisation problem, $\arg\min_{\mathbf{x}} \chi^2(\mathbf{x})$, is called *(generalised) least squares*. Typically, the factors ϕ_k and thus the prediction functions $\hat{\mathbf{z}}_k$ are only defined on subsets of $\mathbf{x}$ as illustrated in the *factor graph* in Figure 2.2(b).

2.3.2 Well-posed Problems and Gauge Freedom

Ideally, a Gaussian state estimation problem is *well-posed* so that the quadratic energy (2.25) has a unique global minimum. In this case, the matrix $\mathtt{J}_{\mathbf{d}}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J}_{\mathbf{d}} \in \mathbb{R}^{n \times n}$ has full rank such that the the normal equation,

$$(\mathtt{J}_{\mathbf{d}}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J}_{\mathbf{d}}) \boldsymbol{\delta} = -\mathtt{J}_{\mathbf{d}}^\top \mathbf{d}(\mathbf{x}) \ , \tag{2.28}$$

has a unique solution. On the other hand, if the normal equation has a p-dimensional solution space, the matrix $\mathtt{J}_{\mathbf{d}}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J}_{\mathbf{d}}$ is singular with rank $n - p$. In this case we say that the state estimation problem has a p-dimensional *gauge freedom*.

2.3.3 Covariance Back-Propagation

So far, we have considered only how to find the single most probable set of parameters $\mathbf{x}$. If there is no gauge freedom in the minimisation of $\chi^2(\mathbf{x})$, then $\mathtt{J}_{\mathbf{d}}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J}_{\mathbf{d}}$ has full rank and we can determine a full distribution for $\mathbf{x}$. A first-order approximation of the information matrix $\Lambda_{\mathbf{x}}$ can be calculated using covariance backpropagation (Hartley & Zisserman, 2004, pp.141):

$$\Lambda_{\mathbf{x}} = \mathtt{J}_{\mathbf{d}}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J}_{\mathbf{d}} \ . \tag{2.29}$$

The covariance $\Sigma_{\mathbf{x}} = \Lambda_{\mathbf{x}}^{-1}$ can be recovered if needed.

2.3.4 Gauss Newton Filter

Let us assume we would like to estimate a parameter $\mathbf{x}$ over time. At each time step $1, ..., t$, we observe a set of measurements $\mathbf{z}_1, ..., \mathbf{z}_t$. We are interested to estimate

the posterior distribution, or belief at time t:

$$bel(\mathbf{x}_t) := p(\mathbf{x}_t|\mathbf{z}_1, ..., \mathbf{z}_t). \tag{2.30}$$

Using the recursive *Bayes filter* scheme (Thrun et al., 2005, Sec. I.2), we get

$$bel(\mathbf{x}_t) \propto p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) bel(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}\ , \tag{2.31}$$

with $p(\mathbf{z}_t|\mathbf{x}_t)$ being the measurement likelihood, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ the prediction probability that $\mathbf{x}_t$ given $\mathbf{x}_{t-1}$ and $bel(\mathbf{x}_{t-1})$ being the prior distribution at time $t-1$. Under the assumption of a *static* parameter (e.g. landmarks in SLAM), i.e. $\mathbf{x}$ does not change over time, it holds that $\int p(\mathbf{x}_t|\mathbf{x}_{t-1}) bel(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1} = bel(\mathbf{x}_{t-1})$. In this case the Bayes filter reduces to

$$bel(\mathbf{x}_t) \propto p(\mathbf{z}_t|\mathbf{x}_t) \cdot bel(\mathbf{x}_{t-1})\ . \tag{2.32}$$

Assuming a Gaussian distribution, $bel(\mathbf{x}_t)$ is proportional to

$$\exp(-(\mathbf{x}_t - \mathbf{x}_{t-1})^\top \Lambda_{\mathbf{x}_{t-1}} (\mathbf{x}_t - \mathbf{x}_{t-1})) \cdot \exp(-(\mathbf{z}_t - \hat{\mathbf{z}}(\mathbf{x}_t))^\top \Lambda_{\mathbf{z}} (\mathbf{z}_t - \hat{\mathbf{z}}(\mathbf{x}_t))). \tag{2.33}$$

Note, that this term is not only the basis of the *Gauss-Newton filter*[3], but also of the correction step of other Gaussian filters such as the popular Extended Kalman Filter (EKF) (Thrun et al., 2005, p.60). As above, maximising the posterior $bel(\mathbf{x}_t)$ is equivalent to minimising the negative log-posterior,

$$\chi^2(\mathbf{x}_t) = (\mathbf{x}_t - \mathbf{x}_{t-1})^\top \Lambda_{\mathbf{x}} (\mathbf{x}_t - \mathbf{x}_{t-1}) + (\mathbf{z}_t - \hat{\mathbf{z}}(\mathbf{x}_t))^\top \Lambda_{\mathbf{z}} (\mathbf{z}_t - \hat{\mathbf{z}}(\mathbf{x}_t)). \tag{2.34}$$

This quadratic energy has two components. The left summand is a *regulariser* which ensures that the state estimate $\mathbf{x}_t$ stays close to the prior distribution $\langle \mathbf{x}_{t-1}, \Lambda_{\mathbf{x}_{t-1}} \rangle$. The right summand is a *data term* which makes sure that the measurement error $\mathbf{z}_t - \hat{\mathbf{z}}(\mathbf{x}_t)$ is minimised. For instance, we can minimise the energy (2.34) by forming the augmented normal equation (2.20) and solve for $\boldsymbol{\delta}$ using Levenberg-Marquardt. The information matrix is updated using uncertainty propagation:

$$\Lambda_{\mathbf{x}_t} = \Lambda_{\mathbf{x}_{t-1}} + \mathrm{J}_{\mathbf{d}_t}^\top \Lambda_{\mathbf{z}} \mathrm{J}_{\mathbf{d}_t}\ . \tag{2.35}$$

The Gauss-Newton filter can also be used to estimate parameters of a *dynamic* system if we assume a *uniform* prior on the motion. Let us assume that the parameter $\mathbf{x} = (\mathbf{a}^\top, \mathbf{b}^\top)^\top$ has a static component $\mathbf{a}$ and a dynamic component $\mathbf{b}$. Thus

[3] The term *Gauss-Newton filter* is borrowed from Sibley et al. (2005). Alternatively, it could be described as an iterated extended information filter *without* state prediction.

the information matrix has the following form:

$$\Lambda_{\mathbf{x}_t} = \begin{bmatrix} \Lambda_{\mathbf{a}_t} & \Lambda_{\mathbf{a}_t,\mathbf{b}_t}^\top \\ \Lambda_{\mathbf{a}_t,\mathbf{b}_t} & \Lambda_{\mathbf{b}_t} \end{bmatrix} . \tag{2.36}$$

After each update (2.35), the dynamic component $\mathbf{b}$ needs to be *marginalised out* (Eustice et al., 2005) from the information matrix:

$$\Lambda'_{\mathbf{a}_t} = \Lambda_{\mathbf{a}_t} - \Lambda_{\mathbf{a}_t,\mathbf{b}_t}^\top \cdot \Lambda_{\mathbf{b}_t}^{-1} \cdot \Lambda_{\mathbf{a}_t,\mathbf{b}_t} . \tag{2.37}$$

Afterwards, the information matrix is augmented with a uniform prior

$$\Lambda'_{\mathbf{x}_t} = \begin{bmatrix} \Lambda'_{\mathbf{a}_t} & 0 \\ 0 & 0 \end{bmatrix} . \tag{2.38}$$

2.4 Lie Groups

The optimisation and filtering methods presented in the previous sections are applicable for scalar fields which are defined on Euclidean vector spaces $\mathbb{R}^n$. When performing optimisation, we calculate an incremental update $\boldsymbol{\delta} \in \mathbb{R}^n$ which is added to the current estimate $\mathbf{x}^{(k)} \in \mathbb{R}^n$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta} \quad \text{(restating equation (2.13))} . \tag{2.39}$$

Now let us consider an expression $G(\boldsymbol{\omega})$. We wish to minimize it with respect to $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$, a rotation in three dimensional space. We can think of $\boldsymbol{\omega}$ as being any parametrisation of rotation in 3D (such as Euler angles or the rotation vector parametrisation defined below equation (2.71) on page 47). Performing a rotation by $\boldsymbol{\delta}$ and then by $\boldsymbol{\omega}$ is in general not equivalent to performing a rotation of $\boldsymbol{\omega} + \boldsymbol{\delta}$. Vector addition is simply not the right operation to concatenate rotations. Thus, rotations (together with their concatenation) cannot be modelled as a Euclidean vector space, but as a *Lie group*.

Lie groups gain more and more popularity among researchers in computer vision and robotics. Most introductory texts, however, are either purely application oriented and merely state relevant properties of some specific Lie groups; or they are sophisticated text books which mainly target a mathematical audience. In an attempt to close this gap, a short tutorial on Lie groups is given which does not only introduce relevant Lie groups such as the group of rotation $\mathbf{SO}(3)$ and group of rigid body motion $\mathbf{SE}(3)$, but also gives some insight on the underlying concepts.

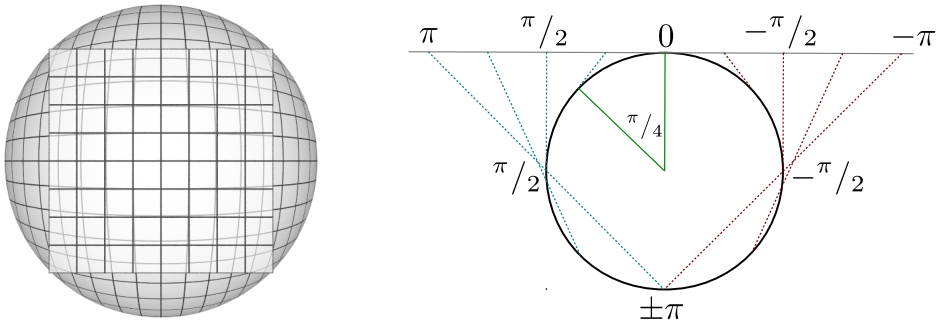


Figure 2.3: (a) The (2-)sphere as an example of a smooth manifold. Locally, it can be approximated by a tangent plane. (b) The circle group (or 1-sphere) represents rotations in a plane. It is a commutative Lie group. A rotation of $\frac{\pi}{4}$ is shown. Concatenating of two rotations is equivalent to adding its angles.

2.4.1 Smooth Manifolds and Lie Groups

Many elementary mathematical methods and results, including the optimisation techniques introduced above, assume that the entities of interest are elements of Euclidean vector spaces. However, most conclusions can be generalised to more abstract concepts. One practical generalisation is the one from Euclidean vector spaces to *(smooth) manifolds*. A smooth manifold is an entity which is locally Euclidean, but might have a different structure globally. Probably the best illustrative example of a manifold is a sphere (see Figure 2.3(a)). Each local area of the sphere can be approximately represented using a tangent plane. However, the global structure is very different from a plane. For instance, while the plane $\mathbb{R}^2$ has no bounds, the sphere is bounded and has a wrap-around. Intuitively, we can regard a manifold as a Euclidean vector space which is bent.

We would like to focus on a special type of manifold which has particular nice properties: *Lie groups*. Formally, a Lie group is a group which is at the same time a smooth manifold. Introducing Lie groups based on this rather abstract definition, however, requires a substantial amount of advanced mathematical theory, such as topology and differential geometry. Instead, we follow the footsteps of Stillwell (2008) and understand Lie groups as closed subgroups of the group of invertible matrices $\mathbf{GL}(n)$, an approach which dates back to von Neumann (1929). Thus, we can avoid a formal introduction of smooth manifolds altogether. Our restriction to

matrix Lie groups is not a major sacrifice since most interesting Lie groups fall into this class.

Lie groups are difficult to introduce using geometric intuition since there is no obvious illustrative example. The sphere, being a great example of a smooth manifold, is not a Lie group.[4] A trivial Lie group is the Euclidean vector space. Obviously, this is not a good example for Lie groups in a similar way that a straight line is not a illustrative example for a smooth function. One of the most basic examples is the circle group (Figure 2.3(b)). This group defines rotation in a plane. Unfortunately, the circle group is not general enough either, since it is commutative. We will see that commutative Lie groups are very special. One of the most simplistic example of a non-commutative Lie group is the group of three dimensional rotations $\mathbf{SO}(3)$. This group will serve as our main example.

2.4.2 Groups

First and foremost, Lie groups are *groups*. A group $(\mathbf{G}, \mathrm{id}, \otimes)$ is a set $\mathbf{G}$ which includes the neutral element $\mathrm{id} \in \mathbf{G}$ together with an operation $\otimes : \mathbf{G} \times \mathbf{G} \to \mathbf{G}$ which fulfils the group axioms:

$$\underset{a\in G}{\forall}\ \underset{b\in G}{\forall}\ \underset{c\in G}{\forall}\quad a \otimes (b \otimes c) = (a \otimes b) \otimes c \quad \text{(associativity)} \tag{2.40}$$

$$\underset{a\in G}{\forall}\quad a \otimes \mathrm{id} = \mathrm{id} \otimes a = a \quad \text{(neutral element)} \tag{2.41}$$

$$\underset{a\in G}{\forall}\ \underset{b\in G}{\overset{=1}{\exists}}\quad a \otimes b = b \otimes a = \mathrm{id} \quad \text{(unique inverse element)} \tag{2.42}$$

From axiom (2.42) it follows directly that each group has an *inverse function* $\mathrm{inv} : \mathbf{G} \to \mathbf{G}$ which can be defined as follow:

$$\mathrm{inv}(a) := \text{ that } b \text{ such that } a \otimes b = \mathrm{id}\ . \tag{2.43}$$

A group is called *commutative* if all elements commute:

$$\underset{a\in G}{\forall}\ \underset{b\in G}{\forall}\ a \otimes b = b \otimes a\ . \tag{2.44}$$

A subset $\mathbf{S} \subset \mathbf{G}$ of a group $(\mathbf{G}, \mathrm{id}, \otimes)$ is called a *subgroup* of $\mathbf{G}$ if $(\mathbf{S}, \mathrm{id}, \otimes)$ is a group. The subgroup $Z(\mathbf{G}) \subset \mathbf{G}$ whose elements commute with all member of $\mathbf{G}$,

$$Z(\mathbf{G}) := \{x \in \mathbf{G} : \underset{y\in\mathbf{G}}{\forall} : x \otimes y = y \otimes x\}\ , \tag{2.45}$$

[4] Indeed, there are only two n-spheres which are Lie groups (for $n \in \mathbb{N}^+$). The 1-sphere/circle, and the 3-sphere/unit quaternions (Stillwell, 2008, p.32).

is called the *centre* (German: Zentrum) of **G**.

2.4.3 Matrix Lie Groups

Let us consider the *general linear group* $\mathbf{GL}(n)$, the most general matrix Lie group. It is the group of invertible $n \times n$ matrices over the real numbers $\mathbb{R}$. The group operation is the matrix multiplication and the neutral element is the identity matrix $\mathtt{I}_{n\times n}$. The matrices must be invertible in order to fulfil axiom (2.42). Note that matrix multiplication, and therefore matrix Lie groups too, are not commutative in general. For example,

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} , \quad \text{but} \quad \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 1 & 1 \end{bmatrix} . \tag{2.46}$$

There are few examples of Lie groups which will be used throughout this tutorial. One is the circle group, or group of in-plane rotations called $\mathbf{SO}(2)$. It consist of 2×2 matrices of the form $\mathtt{R}(\alpha)$,

$$\mathtt{R}(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} . \tag{2.47}$$

If we multiply $\mathtt{R}(\alpha)$ by a 2-vector $\mathbf{x}$, the vector $\mathbf{x}$ is rotated about the origin by angle α anti-clockwise — under the assumption of a right-handed coordinate frame. The Lie group $\mathbf{SO}(2)$ is special because it is commutative. For every two rotations $\mathtt{R}(\alpha)$ and $\mathtt{R}(\beta)$, it does not matter which one is applied first: $\mathtt{R}(\alpha)\mathtt{R}(\beta) = \mathtt{R}(\beta)\mathtt{R}(\alpha)$.

In order to generalise over $\mathbf{SO}(2)$, we first note that its members are orthogonal matrices: $\mathtt{R}(\alpha)\mathtt{R}(\alpha)^\top = \mathtt{I}$; or equivalently: $\mathtt{R}(\alpha)^\top = \mathtt{R}(\alpha)^{-1}$. This can be inferred from $\sin(-\alpha) = -\sin(\alpha)$ and $\cos(-\alpha) = \cos(\alpha)$; therefore $\mathtt{R}(\alpha)^\top = \mathtt{R}(-\alpha) = \mathtt{R}(\alpha)^{-1}$. In general, the group of $n \times n$ matrices with $\mathtt{A}\mathtt{A}^\top = \mathtt{I}$ is called the orthogonal group $\mathbf{O}(n)$. The determinant of an orthogonal matrix is either 1 or -1. Orthogonal matrices with determinant -1 perform a rotation *followed by a reflection*. If we want to achieve pure rotation, we have to restrict ourselves to orthogonal matrices $\mathtt{A}$ with $\det(\mathtt{A}) = 1$. This group is called the *special orthogonal* group $\mathbf{SO}(n)$. We will mainly focus on the group of three dimensional rotations $\mathbf{SO}(3)$.

2.4.4 Tangent Space

Most Lie groups, including $\mathbf{SO}(3)$, are not commutative. However, every group element a commutes with the the identity: $a \otimes id = \mathrm{id} \otimes a$. Thus, if we go *infinitesimally* close to identity, we enter a space which is commutative. This space is the *tangent space* of the Lie group at the identity which we will introduce now.

Definition of smooth path

Let $X \subset \mathbb{R}^m$. Let $[a, b]$ be a real interval.

$$P \text{ is a smooth path in } X \quad :\Leftrightarrow \quad P : [a, b] \to X \text{ is a differentiable function .} \tag{2.48}$$

Thus, a smooth path is a differentiable function from a real interval into a real vector space. Note that whenever necessary, we interpret matrices as m-tuples. Thus, we can consider a square matrix $\mathtt{A}_{n\times n}$ being a member of a vector space $\mathbb{R}^m$ with $m = n^2$. Indeed, we are interested in smooth paths which map into matrix groups. One such path for $\mathbf{SO}(3)$ would be:

$$\mathtt{R}_x : [-\pi, \pi] \to \mathbf{SO}(3), \quad \mathtt{R}_x(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(t) & -\sin(t) \\ 0 & \sin(t) & \cos(t) \end{bmatrix} . \tag{2.49}$$

The term *smooth path* is not arbitrary, but is linked to the concept of motion in a space. Indeed, $\mathtt{R}_x(t)$ describes the rotation around the x-axis at time t. Let us consider a second example: The Lie group $\mathbf{SO}(2)$ is completely determined by the path $\mathtt{R}(\alpha)$ defined in equation (2.47).

Definition: Tangent vector of a path

Let P be a smooth path with $P(0) = \mathbf{y}$.

$$\mathbf{x} \text{ is the tangent vector of path } P \text{ at the point } \mathbf{y} \quad :\Leftrightarrow \quad \mathbf{x} = \frac{\partial}{\partial t} P(t)|_{t=0} \tag{2.50}$$

Moreover, we call $\mathbf{x}$ a tangent vector of a space X if such a smooth path exists.

Definition: Tangent vector of a space

Let $X \subset \mathbb{R}^n$ be a space with $\mathbf{y} \in X$.

$$\begin{aligned} &\mathbf{x} \text{ is a tangent vector of } X \text{ at the point } \mathbf{y} \\ :\Leftrightarrow\ &\exists_{\text{smooth path } P \text{ in } X} \left(P(0) = \mathbf{y} \quad \wedge \quad \mathbf{x} = \tfrac{\partial}{\partial t} P(t)|_{t=0} \right) \ . \end{aligned} \tag{2.51}$$

For example,

$$\frac{\partial}{\partial t} \mathtt{R}_x(t)|_{t=0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin(0) & -\cos(0) \\ 0 & \cos(0) & -\sin(0) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.52}$$

is *the* tangent vector of $\mathtt{R}_x(t)$ and therefore *a* tangent vector of $\mathbf{SO}(3)$. Since it holds that $\mathtt{R}_x(0) = \mathtt{I}$, it is a tangent vector at the identity $\mathtt{I}$.

Furthermore, the set of all tangent vectors at a point $\mathbf{y}$ spans a vector space, the tangent space at $\mathbf{y}$. In particular, the tangent vectors at the identity of a Lie group $\mathbf{G}$ spans a vector space $\mathfrak{g}$, the *tangent spaces at the identity*. If not explicitly stated otherwise, we will assume from now on that tangent vectors and tangent spaces are taken at the identity.

As $\mathtt{R}(\alpha)$ is the only path in $\mathbf{SO}(2)$ and $\mathtt{R}(0) = \mathtt{I}_{2\times 2}$, the tangent vector $\frac{\partial}{\partial \alpha}\mathtt{R}(\alpha)|_{\alpha=0}$,

$$\frac{\partial}{\partial t} \mathtt{R}(\alpha)|_{\alpha=0} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \ , \tag{2.53}$$

spans the tangent space (at the identity) $\mathfrak{so}(2)$.

Finally, let us examine the tangent space of $\mathbf{GL}(n)$. Let $\mathtt{X} \in \mathbb{R}^{n\times n}$ be a general square matrix. It is true that $\mathtt{I} + t\mathtt{X}$ is invertible for a small enough t:

$$\exists_{\epsilon>0}\ \forall_{t\leq\epsilon} : \det(\mathtt{I} + t\mathtt{X}) \neq 0 \ . \tag{2.54}$$

Hence for $t \in [0, \epsilon]$, the path $\mathtt{P}(t) = \mathtt{I} + t\mathtt{X}$ lies within the set of invertible matrices $\mathbf{GL}(n)$. Since it is true that $\mathtt{P}(0) = \mathtt{I}$, $\mathtt{X} = \frac{\partial}{\partial t}(\mathtt{I} + t\mathtt{X})|_{t=0}$ are the tangent vectors of $\mathbf{GL}(n)$ at the identity. Therefore, the tangent space $\mathfrak{gl}(n)$ consists of the set of *all* matrices $\mathtt{X} \in \mathbb{R}^{n\times n}$.[5]

[5] Thanks to Sam L. and Zhen Lin from math.stackexchange.com.

Tangent Space of $\mathbf{O}(3)$ and $\mathbf{SO}(3)$

Now we will construct the tangent space of $\mathbf{O}(3)$ at the identity, which is called $\mathfrak{o}(3)$. Later, we will see that it is identical to the tangent space of $\mathbf{SO}(3)$, thus $\mathfrak{so}(3) = \mathfrak{o}(3)$.

Let us consider a general smooth path $\mathtt{P} : [a, b] \to \mathbf{O}(3)$ with $\mathtt{P}(0) = \mathtt{I}$. Since all such matrices $\mathtt{P}(t)$ are orthogonal, it holds that

$$\mathtt{P}(t) \cdot (\mathtt{P}(t))^\top = \mathtt{I} \ . \tag{2.55}$$

By differentiating this equation on both sides we get (using the product rule):

$$\frac{\partial \mathtt{P}(t)}{\partial t}(\mathtt{P}(t))^\top + \mathtt{P}(t)\frac{\partial (\mathtt{P}(t))^\top}{\partial t} = \mathtt{O} \ . \tag{2.56}$$

Since we are interested in the tangent vector of $\mathtt{P}$, we set $t = 0$ and receive

$$\left.\frac{\partial \mathtt{P}(t)}{\partial t}\right|_{t=0} + \left.\frac{\partial (\mathtt{P}(t))^\top}{\partial t}\right|_{t=0} = \mathtt{O} \ . \tag{2.57}$$

We just showed that the tangent space $\mathfrak{o}(3)$ consists of tangent vectors $\Omega := \frac{\mathtt{P}(t)}{\partial t}|_{t=0}$ such that $\Omega + \Omega^\top = \mathtt{O}$. Hence, $\Omega = -\Omega^\top$ and therefore its diagonal elements have to be zero: $\Omega_{0,0} = \Omega_{1,1} = \Omega_{2,2} = 0$. Furthermore, it is skew-symmetric: $\Omega_{i,j} = -\Omega_{j,i}$. Thus, the tangent space is spanned by the following basis vectors:

$$\mathtt{G}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} , \mathtt{G}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} , \mathtt{G}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \ . \tag{2.58}$$

These basis vectors are sometimes called the (infinitesimal) *generators* of the underlying Lie group. Since there are three linear independent generators for $\mathbf{O}(3)$, the tangent space $\mathfrak{o}(3)$ is three-dimensional. As a side note, this also reveals that group members of $\mathbf{O}(3)$ have three degrees of freedom (DoF). Representing a 3-dimensional space with 9-dimensional basis vectors, written in 3×3-matrix form, might seem to be cumbersome, but allows us to use universal definitions for concepts such as tangent vectors as well as the exponential map and Lie bracket as we will see later. However, a minimal vector representation is sometimes useful, for which we introduce the *hat*-operator $\widehat{\cdot}$,

$$\widehat{\cdot} : \mathbb{R}^m \to \mathbb{R}^{n \times n}, \quad \widehat{\mathbf{x}} = \sum_{k=0}^{m} x_i \mathtt{G}_i, \tag{2.59}$$

which maps a minimal m-vector tangent representation onto an $(n \times n)$ matrix representation. For $\mathfrak{o}(3) = \mathfrak{so}(3)$ the *hat*-operator is:

$$\widehat{\cdot}_{\mathfrak{so}(3)} : \mathbb{R}^3 \to \mathfrak{so}(3), \quad \widehat{\boldsymbol{\omega}}_{\mathfrak{so}(3)} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} := [\boldsymbol{\omega}]_\times \ . \tag{2.60}$$

The notation $[\mathbf{x}]_\times$ is motivated by the fact that $[\mathbf{a}]_\times \cdot \mathbf{b} = \mathbf{a} \times \mathbf{b}$ with $\times$ being the cross product. Sometimes, the inverse function of the hat-operator is required too. For $\mathfrak{o}(3) = \mathfrak{so}(3)$ this *vee-operator* $(\cdot)^\vee$ is:

$$(\cdot)^\vee_{\mathfrak{so}(3)} : \mathfrak{so}(3) \to \mathbb{R}^3 \ , \quad (\Omega)^\vee_{\mathfrak{so}(3)} = \begin{bmatrix} \Omega_{(3,2)} \\ \Omega_{(1,3)} \\ \Omega_{(2,1)} \end{bmatrix} = \begin{bmatrix} -\Omega_{(2,3)} \\ -\Omega_{(3,1)} \\ -\Omega_{(1,2)} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \Omega_{(3,2)} - \Omega_{(2,3)} \\ \Omega_{(1,3)} - \Omega_{(3,1)} \\ \Omega_{(2,1)} - \Omega_{(1,2)} \end{bmatrix} \ . \tag{2.61}$$

2.4.5 Exponential Map

Now we have introduced the tangent space, we need a way to associate elements of it to elements of the underlying Lie group. This is done using the *exponential map*. Let us consider the formal definition of the standard exponential function:

$$e^x : \mathbb{R} \to \mathbb{R}^+, \quad e^x = \sum_{k=0}^{\infty} \frac{x^n}{n!} \ . \tag{2.62}$$

This can be generalized for squared matrices:

$$\exp(\mathtt{X}) : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}, \quad \exp(\mathtt{X}) = \sum_{k=0}^{\infty} \frac{\mathtt{X}^n}{n!} \ , \tag{2.63}$$

with $\mathtt{X}^0 := \mathtt{I}$. Not without reason, this function is called exponential map since it has similar properties to the exponential function including

$$\frac{\partial}{\partial t} \exp(t\mathtt{X}) = \mathtt{X} \exp(t\mathtt{X}) = \exp(t\mathtt{X})\mathtt{X} \ , \tag{2.64}$$

and,

$$\mathtt{XY} = \mathtt{YX} \quad \Rightarrow \quad \exp(\mathtt{X}) \exp(\mathtt{Y}) = \exp(\mathtt{X} + \mathtt{Y}) \ . \tag{2.65}$$

Keep in mind that the latter requires that $\mathtt{X}$ and $\mathtt{Y}$ commute. Furthermore, it can be shown that

$$\exp(\mathtt{AXA}^{-1}) = \mathtt{A} \exp(\mathtt{X}) \mathtt{A}^{-1} \ . \tag{2.66}$$

Corresponding proofs are in Rossmann (2002, pp.2).

In order to develop another important property, we first note that

$$\exp(\mathtt{0}) = \sum_{k=0}^{\infty} \frac{\mathtt{0}^n}{n!} = \mathtt{I} + \mathtt{0} + \mathtt{0} + ... = \mathtt{I} \ . \tag{2.67}$$

Let $\mathtt{X}$ be a general square matrix. It is easy to see that $\mathtt{X}$ and $-\mathtt{X}$ commute; thus

$$\exp(\mathtt{X}) \exp(-\mathtt{X}) \overset{(2.65)}{=} \exp(\mathtt{X} - \mathtt{X}) = \exp(\mathtt{0}) \overset{(2.67)}{=} \mathtt{I} \ , \tag{2.68}$$

hence

$$\exp(\mathtt{X})^{-1} = \exp(-\mathtt{X}) \ . \tag{2.69}$$

We just proved that for all $\mathtt{X} \in \mathbb{R}^{n \times n}$, $\exp(\mathtt{X})$ is invertible. Therefore, $\exp(\cdot)$ maps the set of square matrices into the set of invertible matrices. In other words, it maps element of the tangent space $\mathfrak{gl}(n)$ into the general linear group $\mathbf{GL}(n)$.

Let $\mathbf{G}$ be a closed subgroup of $\mathbf{GL}(n)$ and $\mathfrak{g}$ be the tangent space of $\mathbf{G}$ at the identity. It can be shown (e.g. Stillwell, 2008, pp.143) that

$$\mathtt{X} \in \mathfrak{g} \quad \Rightarrow \quad \exp(\mathtt{X}) \in \mathbf{G} \ . \tag{2.70}$$

In other words, the matrix exponential maps an element from the tangent space to the corresponding matrix Lie group.

Exponential Map onto SO(3)

In order to define the exponential map (2.63) for other matrix Lie groups, we simply have to restrict its domain accordingly. To establish a mapping to $\mathbf{SO}(3)$, we have to restrict the domain on $\mathfrak{o}(3)$ — matrices of the form $[\boldsymbol{\omega}]_\times$. It can be shown that the exponential map $\exp : \mathfrak{o}(3) \to \mathbf{SO}(3)$ is surjective (Gallier, 2011, p.469). This confirms our suspicion that $\mathfrak{o}(3) = \mathfrak{so}(3)$. For $\mathbf{SO}(3)$, the exponential map has a closed form:

$$\exp([\boldsymbol{\omega}]_\times) = \begin{cases} \mathtt{I} + [\boldsymbol{\omega}]_\times + \frac{1}{2}[\boldsymbol{\omega}]_\times^2 = \mathtt{I} & \text{for } (\theta \to 0) \\ \mathtt{I} + \frac{\sin(\theta)}{\theta}[\boldsymbol{\omega}]_\times + \frac{1-\cos(\theta)}{\theta^2}[\boldsymbol{\omega}]_\times^2 & \text{else} \end{cases} \quad \text{with} \quad \theta = ||\boldsymbol{\omega}||_2 \ . \tag{2.71}$$

This mapping is known as the *Rodriguez formula*. It has the following geometric interpretation: A rotation in the three dimensional space can be parametrized using

the so called *rotation vector* $\boldsymbol{\omega}$. Its magnitude defines the rotation angle $\theta = ||\boldsymbol{\omega}||_2$ and the unit vector $\mathbf{u}_{\boldsymbol{\omega}} = \frac{\boldsymbol{\omega}}{\theta}$ the rotation axis. The transformation $\mathtt{R} \cdot \mathbf{x}$ with $\mathtt{R} = \exp(\widehat{\boldsymbol{\omega}})$ rotates a point $\mathbf{x}$ around the axis $\mathbf{u}_{\boldsymbol{\omega}}$ by the angle θ.

2.4.6 Matrix Logarithm

In case that $\exp(\mathfrak{g}) \to \mathbf{G}$ is surjective (=onto) for a given matrix Lie group $\mathbf{G}$, we can define its inverse relation, the *matrix logarithmic*. Even if $\exp(\cdot)$ is not injective, and hence its inverse is a multi-valued function, we typically understand the matrix logarithm as a smooth single-value function $\log : \mathbf{G} \to \mathfrak{g}$ by restricting its codomain accordingly. It holds that

$$\exp(\log(\Omega)) = \Omega \quad \text{and} \quad \log(\exp(\mathtt{A})) = \mathtt{A} \ . \tag{2.72}$$

From property (2.69) it follows that

$$\log(\Omega^{-1}) = -\log(\Omega) \ . \tag{2.73}$$

The matrix logarithm of $\mathbf{SO}(3)$ is

$$\log(\mathtt{R}) = \begin{cases} \frac{1}{2}(\mathtt{R} - \mathtt{R}^\top) = \mathtt{0} & \text{for } d \to 1 \\ \frac{\arccos(d)}{2\sqrt{1-d^2}}(\mathtt{R} - \mathtt{R}^\top) & \text{for } d \in (-1, 1) \end{cases} \quad \text{with} \quad d = \frac{1}{2}(\text{trace}(\mathtt{R}) - 1) \ . \tag{2.74}$$

2.4.7 Adjoint Map

In general, Lie groups are not commutative. Thus, in general $\mathtt{ABA}^{-1} \neq \mathtt{B}$. Hence, it is of interest to ask what $\mathtt{ABA}^{-1}$ equals instead. Let $\mathbf{G}$ be a Lie group and $\mathfrak{g}$ its tangent space. We define a function Ψ:

$$\Psi_{\mathtt{A}} : \mathbf{G} \to \mathbf{G} \ , \quad \Psi_{\mathtt{A}}(\mathtt{B}) := \mathtt{ABA}^{-1} \ , \tag{2.75}$$

with $\mathtt{A} \in \mathbf{G}$. More specifically, we now think of $\mathtt{B}$ being a smooth path through the identity. If we calculate the derivative $\frac{\partial}{\partial t}\Psi_{\mathtt{A}}(\mathtt{B}(t))|_{t=0}$, we receive $\mathtt{AVA}^{-1}$ with $\mathtt{V} := \frac{\partial}{\partial t}\mathtt{B}(t)|_{t=0}$ being a tangent vector in $\mathfrak{g}$. This leads to the definition of the adjoint representation of a Lie group $\mathbf{G}$:

$$\text{Adj}_{\mathtt{A}} : \mathfrak{g} \to \mathfrak{g} \ , \quad \text{Adj}_{\mathtt{A}}(\mathtt{V}) := \mathtt{AVA}^{-1} \ . \tag{2.76}$$

The function $\mathrm{Adj}(\cdot)$ is actually a linear operator. Thus, there exists an $m \times m$ matrix $\mathtt{Ad}_{\mathtt{A}}$ such that

$$\mathtt{Ad}_{\mathtt{A}} : \mathbb{R}^m \to \mathbb{R}^m, \quad \widehat{\mathtt{Ad}_{\mathtt{A}} \cdot \mathtt{x}} = \mathtt{A}\widehat{\mathtt{x}}\mathtt{A}^{-1} \quad (= \mathrm{Adj}_{\mathtt{A}}(\widehat{\mathtt{x}})) \ . \tag{2.77}$$

To get a better understanding of the usefulness of the adjoint, let us apply property (2.66) to the definition above:

$$\widehat{\mathtt{Ad}_{\mathtt{A}} \cdot \mathtt{x}} = \mathtt{A}\widehat{\mathtt{x}}\mathtt{A}^{-1} \quad \overset{(2.66)}{\Rightarrow} \quad \exp(\widehat{\mathtt{Ad}_{\mathtt{A}} \cdot \mathtt{x}}) = \mathtt{A}\exp(\widehat{\mathtt{x}})\mathtt{A}^{-1} \tag{2.78}$$

Thus, $\mathtt{A} \cdot \exp(\widehat{\mathtt{x}}) = \exp(\widehat{\mathtt{Ad}_{\mathtt{A}} \cdot \mathtt{x}}) \cdot \mathtt{A}$, and therefore the adjoint allows us to move the matrix exponential from the right-hand side to the left-hand side of $\mathtt{A}$.

For instance, the adjoint map of $\mathbf{SO}(3)$ is

$$\mathtt{Ad}_{\mathtt{R}} : \mathbb{R}^3 \to \mathbb{R}^3 \ , \quad \mathtt{Ad}_{\mathtt{R}} = \mathtt{R} \ , \tag{2.79}$$

since

$$[\mathtt{R}\mathbf{x}]_\times = \mathtt{R}[\mathbf{x}]_\times \mathtt{R}^\top \ . \tag{2.80}$$

For commutative matrix Lie groups, $\mathtt{Ad} = \mathtt{I}$.

2.4.8 Lie Bracket and Lie Algebra

Let us define the *Lie bracket* for a matrix Lie group $\mathbf{G}$ and its corresponding tangent space $\mathfrak{g}$:

$$[\mathtt{U}, \mathtt{V}] := \mathtt{U}\mathtt{V} - \mathtt{V}\mathtt{U} \ . \tag{2.81}$$

Each such tangent space $\mathfrak{g}$ is closed under the Lie bracket:

$$\mathtt{U} \in \mathfrak{g}, \mathtt{V} \in \mathfrak{g} \quad \Rightarrow \quad [\mathtt{U}, \mathtt{V}] \in \mathfrak{g} \ . \tag{2.82}$$

Thus, a tangent space $\mathfrak{g}$ is not only a vector space, but is also an algebraic structure concerning the Lie bracket and is therefore called *Lie algebra*. The Lie bracket can be derived from the adjoint representation (see Appendix A.3).

It is obvious that $[\mathtt{U}, \mathtt{V}] = \mathtt{0}$ if $\mathtt{U}$ and $\mathtt{V}$ commute. On the other hand for non-commutative elements $\mathtt{U}$ and $\mathtt{V}$, the Lie bracket $[\mathtt{U}, \mathtt{V}]$ "quantifies" how much the commutative law is violated. As mentioned in Section 2.4.4, sometimes it is useful

to represent the Lie algebra using a set of minimal vectors $\mathbf{x}$ instead of a set of square matrices $\widehat{\mathbf{x}}$. In this case, we define the Lie bracket as

$$[\cdot,\cdot] : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m \ , \quad [\mathbf{u},\mathbf{v}] := (\widehat{\mathbf{u}}\widehat{\mathbf{v}} - \widehat{\mathbf{v}}\widehat{\mathbf{u}})^\vee \ . \tag{2.83}$$

For the Lie algebra $\mathfrak{so}(3)$, the Lie bracket on 3-vectors is the cross product:

$$([\mathbf{u}]_\times \cdot [\mathbf{v}]_\times - [\mathbf{v}]_\times \cdot [\mathbf{u}]_\times)^\vee_{\mathfrak{so}(3)} = \mathbf{u} \times \mathbf{v} \ , \tag{2.84}$$

as shown in Appendix A.3.2.

2.4.9 More Examples of Lie Groups

Euclidean Vector Space

Lie groups can be seen as a generalisation of a Euclidean vector space under vector addition. Therefore, the Euclidean vector space $\mathbb{R}^n$ is a trivial example of a Lie group. Note that addition in the Euclidean vector space $\mathbb{R}^n$ can be interpreted as a multiplicative matrix group,

$$\mathbf{c} = \mathbf{a} + \mathbf{b} \quad \Leftrightarrow \quad \begin{bmatrix} \mathtt{I}_{n\times n} & \mathbf{c} \\ \mathtt{0}_{1\times n} & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{I}_{n\times n} & \mathbf{a} \\ \mathtt{0}_{1\times n} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathtt{I}_{n\times n} & \mathbf{b} \\ \mathtt{0}_{1\times n} & 1 \end{bmatrix} \ . \tag{2.85}$$

The tangent space of such matrices is created using the following smooth paths:

$$P_{\mathbf{a}}(t) = \begin{bmatrix} \mathtt{I}_{n\times n} & t\mathbf{a} \\ \mathtt{0}_{1\times n} & 1 \end{bmatrix} \quad \text{with} \quad \mathbf{a} \in \mathbb{R}^n \ . \tag{2.86}$$

Thus, the tangent space consists of matrices of the form

$$\widehat{\mathbf{a}} := \frac{\partial}{\partial t} P_{\mathbf{a}}(t) = \begin{bmatrix} \mathtt{0}_{n\times n} & \mathbf{a} \\ \mathtt{0}_{1\times n} & 0 \end{bmatrix} \ . \tag{2.87}$$

Since $\widehat{\mathbf{a}}$ is nilpotent with $\widehat{\mathbf{a}}^2 = \mathtt{0}$, we get for the matrix exponential and logarithmic map,

$$\exp(\widehat{\mathbf{a}}) = \sum_{k=0}^{\infty} \frac{\widehat{\mathbf{a}}^k}{k!} = \mathtt{I} + \widehat{\mathbf{a}} + \mathtt{0} + \mathtt{0} + \ldots = \begin{bmatrix} \mathtt{I}_{n\times n} & \mathbf{a} \\ \mathtt{0}_{1\times n} & 1 \end{bmatrix} \quad \text{and} \quad \log \begin{bmatrix} \mathtt{I}_{n\times n} & \mathbf{a} \\ \mathtt{0}_{1\times n} & 1 \end{bmatrix} = \widehat{\mathbf{a}} \ . \tag{2.88}$$

Group of Scaling

Let us consider the centre of $\mathbf{GL}(n)$. We are looking for matrices $\mathtt{S}$ with:

$$\underset{\mathtt{X}\in\mathbf{GL}(n)}{\forall} \mathtt{XS} = \mathtt{SX} \ . \tag{2.89}$$

These are matrices of the form $\mathtt{S} = s\mathtt{I}$ with $s \in \mathbb{R} \setminus \{0\}$. One can verify immediately that $(s\mathtt{I})\mathtt{X} = \mathtt{X}(s\mathtt{I})$. Thus, the centre $Z(\mathbf{GL}(n))$ is the group of scaling. If we apply a vector $\mathbf{x}$ to a matrix $s\mathtt{I} \in Z(\mathbf{GL}(n))$, we receive a scaled version of it: $(s\mathtt{I}) \cdot \mathbf{x} = s\mathbf{x}$. Let us now consider the group of *pure* scaling $\mathbb{R}^+(n)$ by restricting ourselves to positive $s \in \mathbb{R}^+$. Obviously, paths in $\mathbb{R}^+(n)$ are of the form

$$\mathtt{P}(s) = s\mathtt{I} \ , \tag{2.90}$$

and hence the one dimensional tangent space is spanned by the generator

$$\mathtt{G}_1 = \mathtt{I} \ . \tag{2.91}$$

The exponential and logarithmic maps are

$$\exp(\sigma\mathtt{I}) = e^{\sigma}\mathtt{I} \ , \quad \text{and} \quad \log(s\mathtt{I}) = \ln(s)\mathtt{I} \ . \tag{2.92}$$

Special Euclidean Group

Finally, we look at the special Euclidean group $\mathbf{SE}(3)$. It is the group of rigid transformation, translation and rotation, in the three dimensional space. A rotation $\mathtt{R} \in \mathbf{SO}(3)$ together with a translation $\mathbf{t} \in \mathbb{R}^3$ is called a *pose*. Such pose transformations are of the following form:

$$\mathtt{R}_{ba}\mathbf{x}_a + \mathbf{t}_{ba} = \mathbf{x}_b \ . \tag{2.93}$$

Pose transformations are best understood if one uses the concept of six DoF reference frames (see Figure 2.4). Let use assume we have two such reference frames a and b. The transformation $\mathtt{R}_{ba}\mathbf{x}_a + \mathbf{t}_{ba}$ transforms a point $\mathbf{x}_a$ in frame a to a point $\mathbf{x}_b$ in reference frame b. Elements of $\mathbf{SE}(3)$ can be written in terms of 4×4 matrices:

$$\begin{pmatrix} \mathbf{x}_b \\ 1 \end{pmatrix} = \mathtt{T}_{ba} \begin{pmatrix} \mathbf{x}_a \\ 1 \end{pmatrix} \quad \text{with} \quad \mathtt{T}_{ba} := \begin{bmatrix} \mathtt{R}_{ba} & \mathbf{t}_{ba} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.94}$$

and thus $\mathbf{SE}(3)$ is a subgroup of $\mathbf{GL}(4)$. It is often convenient to use the shorthand notation $\mathtt{T}_{ba} \cdot \mathbf{x}_a := \mathtt{R}_{ba}\mathbf{x}_a + \mathbf{t}_b$.

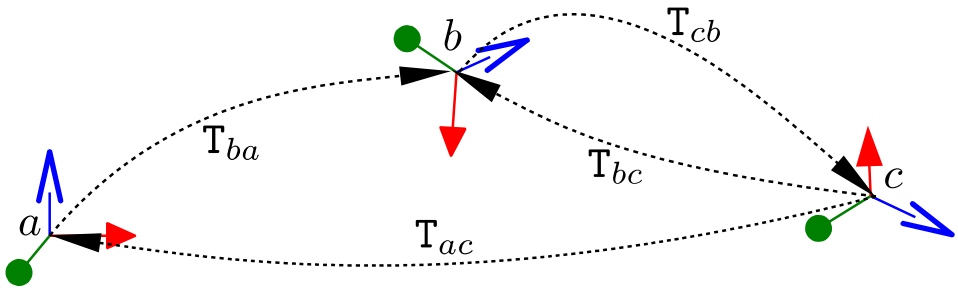


Figure 2.4: Pose transformation between three different reference frames a, b and c. The six DoF frames are represented using three coordinate axis which can be rotated in 3D.

Concatenation of two rigid transformations — a to b, then b to c — is performed using matrix multiplication:

$$\begin{bmatrix} \mathtt{R}_{ca} & \mathbf{t}_{ca} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{R}_{cb} & \mathbf{t}_{cb} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R}_{ba} & \mathbf{t}_{ba} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{R}_{cb}\mathtt{R}_{ba} & \mathtt{R}_{cb}\mathbf{t}_{cb} + \mathbf{t}_{cb} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} . \tag{2.95}$$

Given a transformation $\mathtt{T}_{ba}$, the inverse transformation $\mathtt{T}_{ab}$ is calculated using the matrix inverse:

$$\mathtt{T}_{ab} = \mathtt{T}_{ba}^{-1} = \begin{bmatrix} \mathtt{R}_{ba}^{\top} & -\mathtt{R}_{ba}^{\top}\mathbf{t}_{ba} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} . \tag{2.96}$$

The tangent space $\mathfrak{se}(3)$ is spanned by the generators $\mathtt{G}_i = (\widehat{\mathbf{e}}_i)_{\mathfrak{se}(3)}$ with $\mathbf{e}_i$ being the ith Cartesian unit vector of $\mathbb{R}^6$ and $\widehat{\cdot}_{\mathfrak{se}(3)}$,

$$\widehat{\cdot}_{\mathfrak{se}(3)} : \mathbb{R}^6 \to \mathbb{R}^6 , \quad \widehat{\begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{pmatrix}}_{\mathfrak{se}(3)} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 0 \end{bmatrix} , \tag{2.97}$$

the hat-operator of $\mathfrak{se}(3)$. In order to prove that the tangent space of $\mathbf{SE}(3)$ indeed consists of matrices of the form $\widehat{\mathbf{x}}_{\mathfrak{se}(3)}$, we need to show that $\exp(\widehat{\mathbf{x}}_{\mathfrak{se}(3)}) \in \mathbf{SE}(3)$ for all $\mathbf{x} \in \mathbb{R}^6$ and that $\exp : \mathfrak{se}(3) \to \mathbf{SE}(3)$ is surjective. We use the following lemma from Gallier (2011, pp.479):

Let $\mathtt{X} \in \mathfrak{gl}(n)$, $\mathbf{y} \in \mathbb{R}^n$ and $\mathtt{A} = \begin{pmatrix} \mathtt{X} & \mathbf{y} \\ 0 & 0 \end{pmatrix}$.

$$\exp(\mathtt{A}) = \begin{pmatrix} \exp(\mathtt{X}) & \mathtt{V}\mathbf{y} \\ \mathtt{0}_{1\times n} & 1 \end{pmatrix} \quad \text{with} \quad \mathtt{V} = \sum_{k=0}^{\infty} \frac{\mathtt{X}^k}{(k+1)!} . \tag{2.98}$$

Applying this lemma for $\mathtt{X} = [\boldsymbol{\omega}]_\times$, we get

$$\exp(\boldsymbol{\upsilon}, \boldsymbol{\omega})_{\mathfrak{se}(3)} := \exp\left(\widehat{\begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{pmatrix}}_{\mathfrak{se}(3)}\right) = \begin{pmatrix} \exp([\boldsymbol{\omega}]_\times) & \mathtt{V}\boldsymbol{\upsilon} \\ 0 & 1 \end{pmatrix} \in \mathbf{SE}(3) \ , \tag{2.99}$$

since $\exp([\boldsymbol{\omega}]_\times) \in \mathbf{SO}(3)$ and $\mathtt{V}\boldsymbol{\upsilon} \in \mathbb{R}^3$. It only remains to show that the linear map $\mathtt{V} : \mathbb{R}^3 \to \mathbb{R}^3$ is surjective, or in other words, that the matrix $\mathtt{V}$ is invertible. Again, this proof is given in Gallier (2011, pp.480) together with a closed form solution for $\mathtt{V}$:

$$\mathtt{V} = \begin{cases} \mathtt{I} + \frac{1}{2}[\boldsymbol{\omega}]_\times + \frac{1}{6}[\boldsymbol{\omega}]_\times^2 = \mathtt{I} & \text{for } (\theta \to 0) \\ \mathtt{I} + \frac{1-\cos(\theta)}{\theta^2}[\boldsymbol{\omega}]_\times + \frac{\theta-\sin(\theta)}{\theta^3}[\boldsymbol{\omega}]_\times^2 & \text{else} \end{cases} \quad \text{with} \quad \theta = ||\boldsymbol{\omega}||_2 \tag{2.100}$$

2.4.10 Optimization with respect to Lie Groups

Finally, we can generalize the optimization approaches we introduced in Section 2.2 to functions which work on Lie groups. First, we need to define what we understand by a partial derivatives on Lie groups. For instance, if we consider $\mathbf{SO}(3)$, it becomes clear that expressions of the form $\frac{\partial \mathtt{R}}{\partial R_{(i,j)}}$ are not what we want. Changing a single entry of an orthogonal matrix infinitesimally would make the matrix non-orthogonal and we would leave the space of $\mathbf{SO}(3)$. Elements of $\mathbf{SO}(3)$ have only three DoF. If we want to calculate the derivative at $\mathtt{R}$, there are exactly three Cartesian directions along which we can modify $\mathtt{R}$: the basis vectors of the tangent space at $\mathtt{R}$.

Let us consider a Lie group $\mathbf{G} \subset \mathbf{GL}(n)$, its m-dimensional Lie algebra $\mathfrak{g}$ and and an element $\mathtt{T} \in \mathbf{G}$. The tangent space at $\mathtt{T}$ is spanned by smooth paths of the form $\mathtt{T}_k(t) := \exp(\widehat{t\mathbf{e}_k})\mathtt{T}$ with $\mathbf{e}_k \in \mathbb{R}^m$ being the kth Cartesian unit vector. Using property (2.64), we can verify that

$$\frac{\partial}{\partial \epsilon_k} \exp(\widehat{\boldsymbol{\epsilon}})\bigg|_{\boldsymbol{\epsilon}=\mathbf{0}} := \frac{\partial}{\partial t} \exp(\widehat{t\mathbf{e}_k})\bigg|_{t=0} = \mathtt{G}_k \ , \tag{2.101}$$

with $\mathtt{G}_k$ being the kth generator of $\mathbf{G}$. This is not surprising since it holds for all Lie groups that $\exp(\widehat{t\mathbf{e}_k})$ is a smooth path through the identity; hence its tangent vector is $\mathtt{G}_k$ by definition. Thus, we can define the partial derivative of $\mathtt{T}$ as:

$$\frac{\partial \mathtt{T}_k(t)}{\partial t}\bigg|_{t=0} = \frac{\partial \exp(\widehat{\boldsymbol{\epsilon}})\mathtt{T}}{\partial \epsilon_k}\bigg|_{\boldsymbol{\epsilon}=\mathbf{0}} = \mathtt{G}_k \cdot \mathtt{T} \ . \tag{2.102}$$

More complex derivatives can be calculated using the chain rule (see Appendix B).

Thus, we can perform iterative optimization methods with respect to matrix Lie groups. As an example, let us consider a simple least-squares problem, $F := (\mathbf{d}(\mathtt{T}))^2$. We wish to minimize F with respect to $\mathtt{T} \in \mathbf{G}$. The Jacobian of $\mathbf{d}$ with respect to $\mathtt{T}$ is $\mathtt{J} := \frac{\partial \mathbf{d}(\exp(\widehat{\boldsymbol{\epsilon}})\mathtt{T})}{\partial \epsilon_k}|_{\boldsymbol{\epsilon}=\mathbf{0}}$, hence we get the following normal equation:

$$(\mathtt{J}^\top \mathtt{J})\boldsymbol{\delta} = -\mathtt{J}^\top \mathbf{d}(\mathtt{T}) \ . \tag{2.103}$$

Since the composition of matrix group elements is performed using the matrix multiplication (and not vector addition), we have to modify the iterative update rule (2.13) accordingly:

$$\mathtt{T}^{(k+1)} = \exp\left(\widehat{\boldsymbol{\delta}}\right) \cdot \mathtt{T}^{(k)} \ .^6 \tag{2.104}$$

2.5 Summary

In this chapter, we presented the theoretical foundation which is vital for the remainder of this thesis. In particular, we introduced optimisation techniques, probabilistic state estimation and filtering. Furthermore, we introduced Lie groups as a generalisation over the Euclidean vector space, and showed several examples — including the group of rigid transformations $\mathbf{SE}(3)$. Finally, we showed how optimisation can be applied with respect to Lie groups.

[6] Since we defined the derivative (2.102) using a *left* multiplication of $\mathtt{T}$ by an infinitesimal change $\exp(\widehat{\boldsymbol{\epsilon}})$, we also have to *left* multiply $\mathtt{T}$ by the iterative update $\exp(\widehat{\boldsymbol{\delta}})$.

Chapter 3

Monocular Exploration

In which we present an efficient framework for joint estimation of structure and motion from a single moving camera.

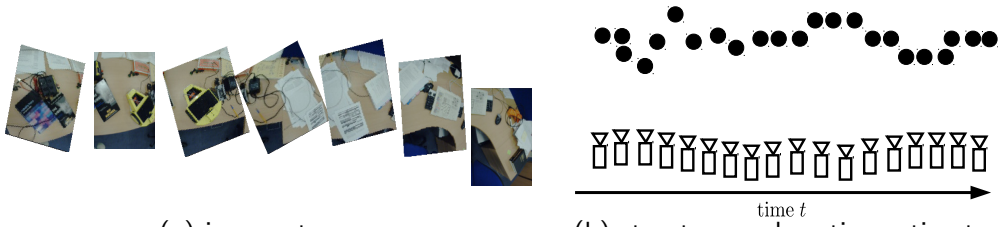
Various stereo SLAM systems have been presented in recent years which are able to provide high-accurate camera motion estimates over large areas, while having only a low computational demand (Konolige & Agrawal, 2008; Mei et al., 2009; Lim et al., 2011). Important application areas in robotics and beyond open up if similar performance can be demonstrated using monocular vision, since a single camera will always be cheaper, more compact and easier to calibrate than a multi-camera rig. However, it has proven more difficult to achieve real-time large-scale visual SLAM with a monocular camera, due to its nature as a purely projective sensor. Geometry does not just 'pop out' of the data from a moving camera, but must be inferred over time from multiple images. Difficulties had to be overcome before a probabilistic sequential approach could successfully be implemented for monocular SLAM due to the fact that image features must be observed from multiple viewpoints with parallax before fully constrained 3D landmark estimates can be made. Special attention was given to feature initialisation schemes which permitted sequential probabilistic estimation of the joint camera and map state (Davison, 2003; Solà et al., 2005; Montiel et al., 2006). These issues, and the fact that monocular maps are just generally less well constrained than those built by metric sensors, meant that it was more difficult than in SLAM systems with other sensors to build algorithms which

could tackle large areas by composing local fragments.

In this chapter we concentrate on the *explorational* aspects of monocular SLAM: The problem of estimating a camera path which is exploring the environment. This concept of exploration is very related to *visual odometry*: a purely incremental motion estimate. However, we will highlight that our approach is more general than visual odometry in a strict sense. In particular, we will talk about the visual front-end (image processing) as well as the optimisation back-end (joint structure and motion estimation). For the back-end, we employ *bundle adjustment* (BA) in a sliding window fashion. Another popular approach is based on Gaussian filters such as the Extended Kalman Filter. The advantages of BA over filtering are quantified in Chapter 4. In this chapter, we do not consider the problem of revisiting known places. In this case, an incremental error in the pose estimate can be corrected using *loop closure correction* techniques which will be discussed in Chapter 5. Even though this chapter is mainly motivated as an introduction to various concepts required later such as camera models, efficient bundle adjustment and feature matching, it contains some novel contributions too: Optical flow guided feature tracking, and a near uniform feature selection scheme using quadtrees. In addition, we will discuss feature initialization using a set of independent filters.

3.1 Monocular SLAM and Exploration

Assume a camera is moving in space and is recording a stream of images as illustrated in Figure 3.1(a). Given these images, we would like to estimate the motion of the camera. Thus, we would like to estimate the path of the camera through the space of rigid transformations $\mathbf{SE}(3)$. At the same time, we would like to estimate the structure of the environment. Typically, we discretise the problem in space and in time. As shown in Figure 3.1(b), we represent the camera path through poses $..., \mathtt{T}_{t-1}, \mathtt{T}_t, \mathtt{T}_{t+1}, ...$ at discrete time steps and the environment using a set of points. This problem of SLAM (Simultaneous Localisation and Mapping) using a monocular camera can be decomposed into two subtasks. The first problem is to associate points or regions in one image with points or regions in a subsequent images. While performing this feature tracking, it is not only important to account for appearance similarities among the images, but also make sure that these feature tracks mutually agree so they can be explained by the camera motion. A framework



(a) image stream (b) structure and motion estimate

Figure 3.1: Monocular exploration

which fulfils this task is called a *visual front-end*. The second task is to jointly estimate camera motion and structure given the visual feature associations from the front-end. This task is tackled by the *optimisation back-end* which estimates the structure and motion parameters by minimising the distance between the projections of the estimated 3D points onto the image and their visual measurements.

In this chapter we focus on estimating a camera path while the camera is exploring the environment. The challenge is to receive an accurate estimate of large scale motion while keeping the computational cost bounded and thus achieve (near) real-time performance. The explorational aspect of monocular SLAM is related to visual odometry. The concept of *odometry* comes from the field of *mobile robotics* where wheel encoders measure the rotations of the individual wheels of a vehicle. This leads to incremental pose estimate where the current pose T_t of the robot only depends on the previous pose T_{t-1} and the corresponding odometry measurement $\mathbf{u}_{t-1,t}$. Therefore, visual odometry in a strict sense refers to incremental pose estimation where we estimate the current pose T_t given the previous one T_{t-1} and visual measurements $\mathcal{Z}_{t-1,t}$ from the temporally adjacent image pair I_{t-1}, I_t. Instead we describe our approach as *monocular exploration* in order to emphasize that this strict definition of visual odometry shall be relaxed. By allowing the integration of measurements from non-consecutive frames, we can continue to track visual features which are temporarily occluded and therefore can deal with mini loop closures. Furthermore, we relax the concept of temporally consecutive frames to keyframes which are sampled in the spatial domain.

Before describing the optimisation back-end and visual front-end of the monocular SLAM framework in detail, we need to introduce the camera model.

3.2 Camera Model

3.2.1 Monocular Camera Model

Let us assume we have a camera which provides us with a stream of images. Furthermore, let us denote image coordinates as $\mathbf{z} = (u, v)$. We follow the common convention that the top-left corner is the origin $\mathbf{0}$ with the u-axis pointing right and the v-axis down. In the forward model, we would like to project points in the world $\mathbf{y}$ onto the image plane I. Without loss of generality, we assume we have the following right-handed three dimensional coordinate system: The y_1-axis points right, the y_2-axis points down and the y_3-axis points forward. The physical pose of the camera in the world is described by a rigid point transformation $\mathtt{T}_{cw} \in \mathbf{SE}(3)$. Thus, in a first step we transform the point $\mathbf{y}$ from the world frame into the camera frame:

$$\mathbf{x} = \mathtt{T}_{cw} \cdot \mathbf{y} = \mathtt{R}_{cw}\mathbf{y} + \mathbf{t}_{cw} \ . \tag{3.1}$$

Afterwards, we employ the well-known pinhole camera model. Let $\mathrm{proj}(\cdot)$ be the function which projects a point $\mathbf{x}$ onto the (x_1, x_2)-plane:

$$\mathrm{proj}(\mathbf{x}) = \frac{1}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \ . \tag{3.2}$$

If f is the focal length and $\mathbf{p} \in I$ the principal point, the pinhole camera model is given by

$$\mathrm{proj}(\mathtt{K} \cdot \mathbf{x}) = f \cdot \mathrm{proj}(\mathbf{x}) + \mathbf{p} \quad \text{with} \quad \mathtt{K} = \begin{bmatrix} f & 0 & p_1 \\ 0 & f & p_2 \\ 0 & 0 & 1 \end{bmatrix} \ , \tag{3.3}$$

where $\mathtt{K}$ is called the intrinsic camera matrix. Putting this together, we receive the following monocular forward model $\hat{\mathbf{z}}_{\mathrm{mono}}$:

$$\hat{\mathbf{z}}_{\mathrm{mono}}(\mathtt{T}_{cw} \cdot \mathbf{y}) := \mathrm{proj}(\mathtt{K} \cdot \mathtt{T}_{cw}(\mathbf{y})) \ . \tag{3.4}$$

This forward equation does not model lens distortion, such as radial distortion. For all algorithms we discuss in this thesis, we assume that the camera images I were undistorted in a preprocessing step. This can be done very efficiently, e.g. using the OpenCV library.[1]

[1] http://code.opencv.org

For monocular cameras, the inverse model is fundamentally more complicated. Let us recapitulate that a point in the world can be associated with a point on the image plane using the forward model $\hat{\mathbf{z}}_{\text{mono}}(\mathtt{T}_{cw} \cdot \mathbf{y})$. However due to the projective nature of a monocular camera, a measured point $\mathbf{z} = (u, v)^\top$ on the image plane can only be associated with a infinite ray in the world:

$$\mathbf{y}(x_3) = \mathtt{T}_{cw}^{-1} \begin{pmatrix} \frac{x_3}{f}(u - p_1) \\ \frac{x_3}{f}(v - p_2) \\ x_3 \end{pmatrix} , \tag{3.5}$$

where x_3 is the unknown depth. Therefore given two camera views with known relative displacement $\mathtt{T}_{ba} = [\mathtt{R}_{ba}, \mathbf{t}_{ba}]$, a point in frame a corresponds to a line in frame b. In order to concretise this, let us note that a 2D point $\mathbf{z} = (z_1, z_2)^\top$ can be represented using a 3-vector such that $(z_1, z_2, 1)^\top = (kz_1, kz_2, k)^\top$ with $k \neq 0$. This representation is called the *homogeneous* representation of a point. Two representations, (a_1, a_2, a_3) and (b_1, b_2, b_3), are equivalent if a factor $k \neq 0$ exists such that $(a_1, a_2, a_3) = (kb_1, kb_2, kb_3)$. Furthermore, a 2D line can be defined using a homogeneous representation $\mathbf{l} = (l_1, l_2, l_3)^\top$, such that all homogeneous points $\mathbf{x}$ on the line fulfil the equation $x_1 l_1 + x_2 l_2 + x_3 l_3 = 0$. In the field of two-view geometry, it is well known that a (homogeneous) point $\mathbf{z}_a$ in frame a corresponds to the (homogeneous) line $\mathbf{l}_b$,

$$\mathbf{l}_b = \mathtt{E} \cdot \mathbf{z}_a \ , \tag{3.6}$$

in frame b; $\mathtt{E} := \mathtt{K} \cdot \mathtt{R}_{ba} \cdot [\mathbf{t}_{ba}]_\times \cdot \mathtt{K}^{-1}$ is called the *essential matrix* and $\mathbf{l}_b$ is called the *epipolar line* (Hartley & Zisserman, 2004, chap. 9). A point $\mathbf{z}_b$ lies on the epipolar line $\mathbf{l}_b$ if and only if it holds that $\mathbf{z}_b^\top \mathbf{l}_b = 0$. Thus, for a point correspondence pair $\mathbf{z}_a \leftrightarrow \mathbf{z}_b$ the following epiploar constraint,

$$\mathbf{z}_b^\top \cdot \mathtt{E} \cdot \mathbf{z}_a = 0 \ , \tag{3.7}$$

must hold. Due to its homogeneous formulation, this constraint is invariant to the scale $s \neq 0$ of the translation $\mathbf{t}_{ba}$, which is easy to verify by $\mathbf{z}_b^\top \cdot \mathtt{KR}_{ba}[s\mathbf{t}_{ba}]_\times \mathtt{K}^{-1} \cdot \mathbf{z}_a = \mathbf{z}_b^\top \cdot \mathtt{KR}_{ba}[\mathbf{t}_{ba}]_\times \mathtt{K}^{-1} \cdot (s\mathbf{z}_a) = \mathbf{z}_b^\top \cdot \mathtt{KR}_{ba}[\mathbf{t}_{ba}]_\times \mathtt{K}^{-1} \cdot \mathbf{z}_a$. However, it has a singularity for $\mathbf{t}_{ba} = \mathbf{0}$ since in that case equation (3.7) is always true. This singularity is illustrated in Figure 3.2.

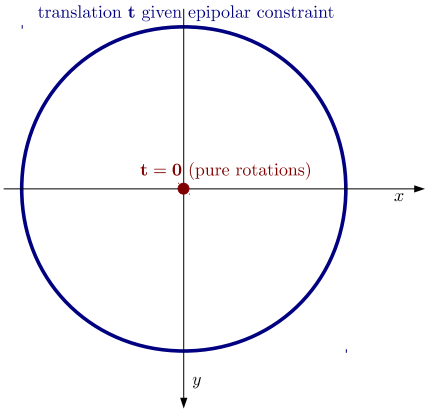


Figure 3.2: We can assume without loss of generality that all possible translations describable by essential matrices $\mathtt{E}$ lie on the unit sphere, since the epipolar constraint $\mathbf{z}_j^\top \cdot \mathtt{E} \cdot \tilde{\mathbf{z}}_j = \mathbf{0}$ has a scale ambiguity. However, $\mathtt{E}$ is undefined for zero translation $\mathbf{t} = \mathbf{0}$. In other words, the space of pure rotations is *not* smoothly connected to the space of rotations and translations modulo scale (defined by the epipolar constraint).

3.2.2 Stereo Camera Model

Even thought we focus on monocular SLAM in this chapter, for completeness we shall present the stereo camera model as well. By stating also the stereo formulation of SLAM, we can highlight the unique characteristics of the monocular case even better. So, let us assume we have a stereo camera rig. A point in the world might projects in the left image I_l as well as the right image I_r. Furthermore, let us assume that the images are *rectified*: We can assume that both images lie in the same plane and that a row in image I_r corresponds to a row in image I_l. Thus, it is true that $v_l = v_r$ given that (u_l, v_l) and (u_r, v_r) are observations of the same physical point in the world. Therefore, a stereo observation can be represented using a 3-vector $\mathbf{z} = (u_l, v_l, u_r)^\top$. Furthermore, we assume we know the horizontal offset between the origin in I_l and the origin in I_r in metres. This offset is called *baseline* b. Now, we have enough information to specify the stereo forward model:

$$\hat{\mathbf{z}}_{\text{stereo}}(\mathtt{T}_{lw} \cdot \mathbf{y}) := \begin{pmatrix} \hat{\mathbf{z}}_{\text{mono}}(\mathtt{T}_{lw} \cdot \mathbf{y}) \\ f\frac{x_1 - b}{x_3} + p_1 \end{pmatrix} = \begin{pmatrix} f \cdot \text{proj}(\mathbf{x}) + \mathbf{p} \\ f\frac{x_1 - b}{x_3} + p_1 \end{pmatrix} \quad \text{with} \quad \mathbf{x} := \mathtt{T}_{lw} \cdot \mathbf{y} \ . \tag{3.8}$$

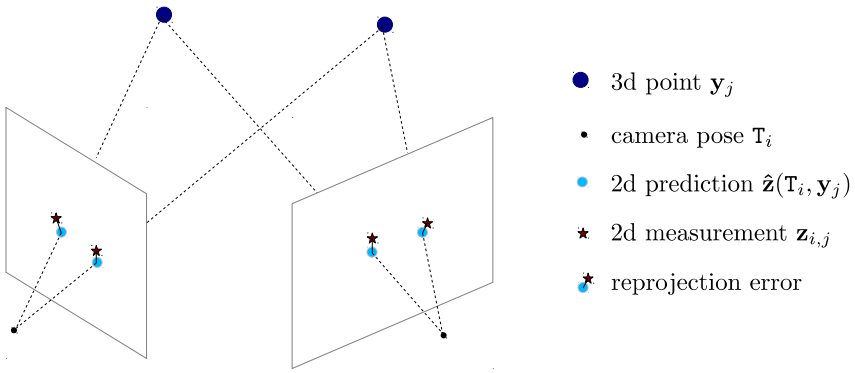


Figure 3.3: In bundle adjustment, we optimise over the set of 3D points $\mathbf{y}_i$ and camera poses $\mathtt{T}_j$ by minimising the reprojection errors $|\mathbf{z}_{i,j} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}_j)|$.

For stereo cameras, the inverse model is:

$$\mathbf{y} = \mathtt{T}_{lw}^{-1} \begin{pmatrix} \frac{x_3}{f}(u_l - p_1) \\ \frac{x_3}{f}(v_l - p_2) \\ x_3 \end{pmatrix} \quad \text{with} \quad x_3 = \frac{b}{f(u_l - u_r)} \,. \tag{3.9}$$

Thus, in contrast to monocular SLAM, the inverse model is straightforward. While a monocular measurement (u, v) can only be associated with an infinite ray, the stereo measurement (u_l, v_l, u_r) has a one to one relation to a point $\mathbf{y}$ in the world. However, this inversion can only be done under the assumption that the measurement (u_l, v_l) in the left frame is associated with the corresponding measurement u_r in the right frame.

3.3 Optimization Back-end

3.3.1 Introduction to Bundle Adjustment

Let us assume a camera is moving in space and recording a sequence of images $I_1, I_2, ..., I_t$. In visual SLAM, we would like to estimate the camera motion $\mathtt{T}_1, \mathtt{T}_2, ..., \mathtt{T}_t$ as well as the scene geometry represented using a set of discrete points $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_j$ as illustrated in Figure 3.1. For now, we assume that the data association problem is solved. Thus, we assume we have a set of observations $\mathcal{Z}$ given, where $\mathbf{z}_{i,j} \in \mathcal{Z}$ is a measurement of point $\mathbf{y}_i$ in frame $\mathtt{T}_j$. In the optimisation back-end, we perform a joint estimation over a set of poses $\mathtt{T}_i$ and a set of points $\mathbf{y}_j$ called bundle adjustment

(Triggs et al., 1999). In bundle adjustment (BA), we estimate these parameters by minimizing the distance $\mathbf{d}_{i,j}$ between point prediction $\hat{\mathbf{z}}(\mathtt{T}_j \cdot \mathbf{y}_i)$ and its measurement $\mathbf{z}_{i,j}$:

$$\mathbf{d}_{i,j}(\mathtt{T}_j, \mathbf{y}_i) := \mathbf{z}_{i,j} - \hat{\mathbf{z}}(\mathtt{T}_j \cdot \mathbf{y}_i) \ . \tag{3.10}$$

This is illustrated in Figure (3.3). Under a Gaussian assumption, this joint problem can be formulated using the following cost function:

$$\chi^2 = \mathbf{d}^\top \Lambda_{\mathbf{z}} \mathbf{d} = \begin{pmatrix} \mathbf{d}_{1,1}^\top & \dots & \mathbf{d}_{m,1}^\top & \mathbf{d}_{1,2}^\top & \dots & \mathbf{d}_{m,n}^\top \end{pmatrix} \Lambda_{\mathbf{z}} \begin{pmatrix} \mathbf{d}_{1,1} \\ \vdots \\ \mathbf{d}_{m,1} \\ \mathbf{d}_{1,2} \\ \vdots \\ \mathbf{d}_{m,n} \end{pmatrix} , \tag{3.11}$$

where Λ is the inverse covariance of the underlying measurement model. We will elaborate on the structure of Λ later.

In order to solve this problem using a Gauss-Newton type method, one could calculate the Jacobian $\mathtt{J}$ of $\mathbf{d}$,

$$\begin{aligned} &\mathtt{J} = \begin{pmatrix} \mathtt{J}_\mathtt{T} & \mathtt{J}_\mathbf{y} \end{pmatrix} \\ \text{with} \quad &\mathtt{J}_\mathtt{T} := \begin{pmatrix} \frac{\partial \mathbf{d}(\exp(\widehat{\mathbf{x}})\mathtt{T}_1, \dots, \mathtt{T}_m, \mathbf{y}_1, \dots, \mathbf{y}_n)}{\partial \mathbf{x}}|_{\mathbf{x}=0} & \dots & \frac{\partial \mathbf{d}(\mathtt{T}_1, \dots, \exp(\widehat{\mathbf{x}})\mathtt{T}_m, \mathbf{y}_1, \dots, \mathbf{y}_n)}{\partial \mathbf{x}}|_{\mathbf{x}=0} \end{pmatrix} \\ &\mathtt{J}_\mathbf{y} := \begin{pmatrix} \frac{\partial \mathbf{d}(\mathtt{T}_1, \dots, \mathtt{T}_m, \mathbf{y}_1, \dots, \mathbf{y}_n)}{\partial \mathbf{y}_1} & \dots & \frac{\partial \mathbf{d}(\mathtt{T}_1, \dots, \mathtt{T}_m, \mathbf{y}_1, \dots, \mathbf{y}_n)}{\partial \mathbf{y}_n} \end{pmatrix} \end{aligned} \tag{3.12}$$

with respect to all poses and points. Afterwards, we can set up the normal equation:

$$\mathtt{J}^\top \Lambda_{\mathbf{z}} \mathtt{J} \boldsymbol{\delta} = -\mathtt{J} \Lambda_{\mathbf{z}} \mathbf{d} \ . \tag{3.13}$$

3.3.2 Gauge Freedom and Monocular Scale Ambiguity

If one proceeds exactly as explained above, one would realise that the matrix $\mathtt{J}^\top \Lambda_{\mathbf{z}} \mathtt{J}$ is singular. This is caused by our decision to optimise over *all* poses $\mathtt{T}_1, ..., \mathtt{T}_m$ and *all* points $\mathbf{y}_1, ..., \mathbf{y}_n$. To get a better understanding of this, let us first consider bundle adjustment using a calibrated stereo rig; i.e. the observations $\mathbf{z}$ are three dimensional and the prediction function is $\hat{\mathbf{z}}_{\text{stereo}}$. Now, let us assume that $\bar{\mathtt{T}}_1, ...,$ $\bar{\mathtt{T}}_m, \bar{\mathbf{y}}_1, ..., \bar{\mathbf{y}}_n$ is the optimal solution to the BA minimization problem where the reprojection error of each point in each frame is minimal. It is important to note

that all point measurements are relative, so that the solution does not depend on the absolute frame of reference. If we were now to apply a general rigid body transformation $\mathtt{A} \in \mathbf{SE}(3)$ to all parameters, the reprojection error would still be minimal. In other words, $\mathtt{A}\bar{\mathtt{T}}_1, ..., \mathtt{A}\bar{\mathtt{T}}_m, \mathtt{A}\bar{\mathbf{y}}_1, ..., \mathtt{A}\bar{\mathbf{y}}_n$ is another optimal solution to the problem. Therefore, we do have a six dimensional solution space, and thus we say that there is a *gauge freedom* of six dimensions. That is the reason why the approximated Hessian is rank deficient. We can remove the gauge freedom by fixing six parameters, e.g. the first pose. Thus, we would optimize over the poses $\mathtt{T}_2, ..., \mathtt{T}_m$ and all points $\mathbf{y}_1, ..., \mathbf{y}_n$ while keeping $\mathtt{T}_1$ fixed which now defines the reference frame.

Let us assume we have done that, but now consider monocular BA; $\mathbf{z}$ is two-dimensional and the prediction function is $\mathbf{z}_{\text{mono}}$. If we optimize over $\mathtt{T}_2, ..., \mathtt{T}_m$, $\mathbf{y}_1, ..., \mathbf{y}_n$ there remains a one dimensional gauge freedom. This is because a monocular camera is an angular measuring device, but it cannot measure distances directly. If one knows the absolute baseline between two frames, one could measure absolute distance using triangulation. However, since we estimate structure and motion jointly, an overall scale ambiguity remains: The set of parameters $[\mathtt{R}_2, \mathbf{t}_2], ...,$ $[\mathtt{R}_m, \mathbf{t}_m], \mathbf{y}_1, ..., \mathbf{y}_n$ leads to exactly the same reprojection error as the scaled set $[\mathtt{R}_2, s\mathbf{t}_2], ..., [\mathtt{R}_m, s\mathbf{t}_m], s\mathbf{y}_1, ..., s\mathbf{y}_n$ for $s \neq 0$ (see also Figure 1.2, p.17). Thus, in monocular BA we need to fix seven parameters in order to remove the Gauge freedom.

3.3.3 Constant-time Bundle Adjustment

If SLAM is performed over time, the number of parameters — frames and points — keeps constantly increasing. Thus, if we were to perform bundle adjustment on all these parameters, the computational cost of a single iteration is unbounded. However, if we wish to perform SLAM in real-time, we need a *constant-time* algorithm such that the cost of a single iteration does not exceed a certain threshold. To achieve this, we need to restrict the number of parameters in the optimisation. It is natural to restrict the number of frames involved, which leads to a restriction on the number of poses and points. There are two common approaches to select a subset of frames. In a pure visual odometry application one can simply select the last m frames and therefore apply BA in a sliding window fashion. Another approach is to sample *keyframes* from the whole trajectory such that they are approximate

uniformly distributed over the explored space. A typical heuristic to achieve this is to only add a new keyframe to the back-end if the distance to the closest keyframe exceeds a threshold. This keyframe approach achieves constant-time performance only if the area of operation is bounded.

Since our focus is monocular exploration, we use a combination of both approaches: As in Mouragnon et al. (2006), we perform BA in a sliding window over a number of spatially separated keyframes. Thus, we only add a new keyframe to our sliding window once the camera has moved significantly far away from the previous keyframe. In this way we can deal with large scale exploration. A pure sliding window approach might have problems with varying camera motion. Especially, it is likely to fail if the camera stays stationary for some time: If there is no significant translation among all frames in the sliding window, no scene depth can be observed. In order to fix the monocular scale ambiguity and anchor the sliding window to the previous poses in the trajectory, we fix the first two keyframes $\mathtt{T}_1$ and $\mathtt{T}_2$ during the optimisation. Even though we fix 12 parameters and therefore five more than necessary, this does not introduce a significant bias in practise. Since $\mathtt{T}_1$ and $\mathtt{T}_2$ were optimised in the previous iterations, we can assume that their relative pose is near optimal. The translation between $\mathtt{T}_1$ and $\mathtt{T}_2$ defines the relative scale which will remain fixed during optimisation (see also Figure 3.4(a)).

3.3.4 Efficient Solution of the Normal Equation

Let us now revisit the normal equation (3.13). First we note that the Jacobian $\mathtt{J}$ looks slightly different now, since we do not optimize over $\mathtt{T}_1$ and $\mathtt{T}_2$: The first two column blocks of $\mathtt{J}_\mathtt{T}$ are missing (i.e. $2 \cdot 6 = 12$ columns). Under the assumption that there are m frames, n poses, and p observations in the sliding window, the pose Jacobian is a $6(m-2) \times 2p$ matrix and the point Jacobian is a $3n \times 2p$ matrix. Therefore, we have a $(6(m-2)+3n) \times 2p$ Jacobian matrix $\mathtt{J}$ and the Hessian $\mathtt{J}^\top \cdot \mathtt{J}$ is a $(6(m-2)+3n) \times (6(m-2)+3n)$ matrix. Therefore a naive implementation of BA would have a computational cost of $O((m+n)^3)$ — under the assumption of a cubic complexity to solve the linear system.[2] Even though we restricted the

[2]The standard implementation of (dense) matrix multiplication and therefore (dense) matrix solvers is cubic in the number of rows/columns m. In 1971, Schönhage & Strassen (1971) presented a faster matrix multiplication/inversion with a complexity of $\mathcal{O}(m^{2.807})$ — with the price of slightly reduced numerical stability (Highham, 1990). When talking about cubic complexity for matrix inversion, we keep in mind that practical algorithms exists which are actually slightly

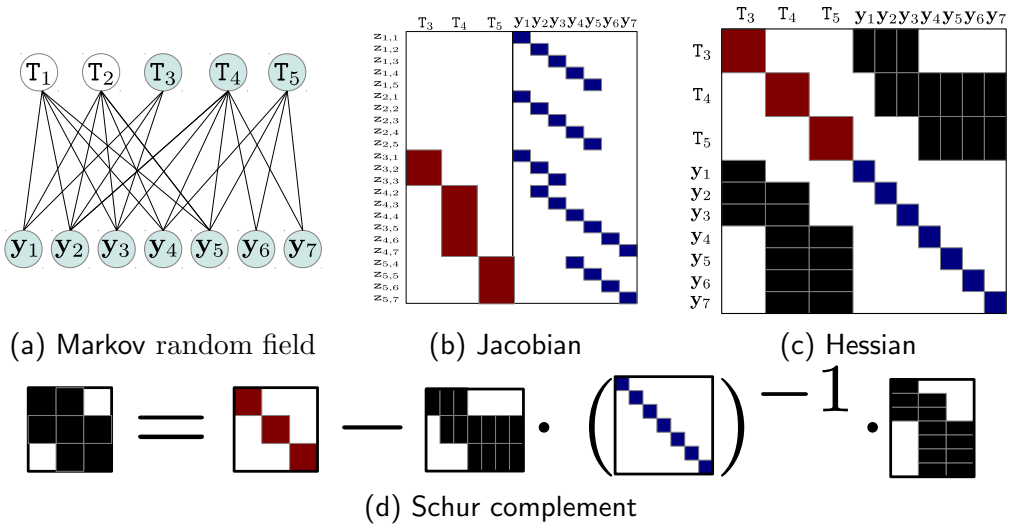


Figure 3.4: Bundle adjustment toy example with five poses and seven points. (a) illustrates this estimation problem using a *Markov Random Field* (MRF). An MRF can be seen as a bipartite factor graph (Figure 2.2), thus with only binary constraints, which represent the measurement $\mathbf{z}_{i,j}$ of point $\mathbf{y}_j$ from pose $\mathtt{T}_i$. The first two poses $\mathtt{T}_1, \mathtt{T}_2$ are fixed in order to remove the Gauge freedom of monocular BA. Strictly speaking, constraints linked to one of the two fixed poses are unary since they only depend on one variable: the point $\mathbf{y}_j$. (b) shows the sparse pattern of the corresponding Jacobian. Each row represents an observation $\mathbf{z}_{i,j}$. Binary constraints result in two non-zero blocks per row, while unary constraints have one non-zero block. (c) shows the sparse Hessian. An off-diagonal block is non-zero if the corresponding point $\mathbf{y}_j$ is visible in frame $\mathtt{T}_i$. (d) illustrates the calculation of the Schur complement.

number of keyframe, this optimization would be quite inefficient since there are typically hundreds or even thousands of points within the sliding window. However if the errors of the individual measurements $\mathbf{z}_{i,j}$ are independent, which is commonly assumed when measurements are made from a calibrated sensor such as a camera with known intrinsics and the energy simplifies to

$$\chi^2 = \sum_{\mathbf{d}_{i,j}} (\mathbf{d}_{i,j}(\mathtt{T}_i, \mathbf{y}_j))^\top \Lambda_{\mathbf{z}_{i,j}} (\mathbf{d}_{i,j}(\mathtt{T}_i, \mathbf{y}_j)) \ . \tag{3.14}$$

In this case, the measurement inverse covariance $\Lambda_{\mathbf{z}}$ is block-diagonal and $\mathtt{J}$ is sparse. As illustrated in Figure 3.4(b), $\mathtt{J}_{\mathtt{T}}$ and $\mathtt{J}_{\mathbf{y}}$ have at most one non-zero block in each row

faster. According to Williams (2011), the theoretical asymptotic complexity is believed to be even quadratic in m, while the most efficient algorithm to date has a complexity of $\mathcal{O}(m^{2.3727})$. However, algorithms of this class have such a large overhead that they would only pay off for astronomically large matrices which are far beyond any practical use.

(= per measurement). Thus, the matrices $\mathtt{J}_\mathtt{T}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathtt{T}$ and $\mathtt{J}_\mathbf{y}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{y}$ are block-diagonal (Figure 3.4(c)). This reflects the fact that there are only binary pose-point constraints, but no pose-pose, point-point or higher order constraints. This special structure of the Hessian leads to the following algebraic trick.

3.3.5 The Schur Complement Trick

Let us consider the invertible block matrix

$$\mathtt{M} = \begin{bmatrix} \mathtt{A} & \mathtt{U} \\ \mathtt{V} & \mathtt{B} \end{bmatrix} , \tag{3.15}$$

with $\mathtt{B}$ being invertible too. If we like to solve the linear equation system

$$\begin{bmatrix} \mathtt{A} & \mathtt{U} \\ \mathtt{V} & \mathtt{B} \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} , \tag{3.16}$$

it is equivalent to solving the following linear system instead:

$$(\mathtt{A} - \mathtt{U}\mathtt{B}^{-1}\mathtt{V})\mathbf{x} = \mathbf{a} - \mathtt{U}\mathtt{B}^{-1}\mathbf{b} . \tag{3.17}$$

This results from pre-multiplying the second line of the system (3.16) by $\mathtt{U}\mathtt{B}^{-1}$ and subtracting it from the first line. The matrix $\mathtt{A} - \mathtt{U}\mathtt{B}^{-1}\mathtt{V}$ is called the *Schur complement* of $\mathtt{M}$. After solving for $\mathbf{x}$, we can solve for $\mathbf{y}$ using back-substitution:

$$\mathbf{y} = \mathtt{B}^{-1}(\mathbf{b} - \mathtt{V}\mathbf{x}) . \tag{3.18}$$

Rewriting the normal equation (3.13) in this form, thus

$$\mathtt{A} = \mathtt{J}_\mathtt{T}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathtt{T} , \tag{3.19}$$

$$\mathtt{B} = \mathtt{J}_\mathbf{y}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{y} , \tag{3.20}$$

$$\mathtt{U} = \mathtt{V}^\top = \mathtt{J}_\mathtt{T}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{y} , \tag{3.21}$$

$$\mathbf{a} = -\mathtt{J}_\mathtt{T} \Lambda_\mathbf{z} \mathbf{d}_\mathtt{T} , \tag{3.22}$$

$$\text{and} \quad \mathbf{b} = -\mathtt{J}_\mathbf{y} \Lambda_\mathbf{z} \mathbf{d}_\mathbf{y} , \tag{3.23}$$

leads to the following solution scheme:

$$(\mathtt{J}_\mathtt{T}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathtt{T} - \mathtt{J}_\mathtt{T}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{y} (\mathtt{J}_\mathbf{y}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{y})^{-1} \mathtt{J}_\mathbf{y}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathtt{T})\mathbf{x} = -\mathtt{J}_\mathtt{T} \Lambda_\mathbf{z} \mathbf{d}_\mathtt{T} + \mathtt{J}_\mathtt{T}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{y} (\mathtt{J}_\mathbf{y}^\top \Lambda_\mathbf{z} \mathtt{J}_\mathbf{y})^{-1} \mathtt{J}_\mathbf{y} \Lambda_\mathbf{z} \mathbf{d}_\mathbf{y} \tag{3.24}$$

$$\mathbf{y} = -(\mathtt{J}_\mathbf{y}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathbf{y})^{-1} (\mathtt{J}_\mathbf{y} \mathtt{\Lambda}_\mathbf{z} \mathbf{d}_\mathbf{y} + \mathtt{J}_\mathbf{y}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathtt{T} \mathbf{x}) \ . \tag{3.25}$$

Since $\mathtt{J}_\mathbf{y}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathbf{y}$ is block-diagonal, we can perform the inversion $(\mathtt{J}_\mathbf{y}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathbf{y})^{-1}$ in linear time (wrt. the number of points n). The solution of equation (3.24) is typically dominated either by the outer product $\mathtt{J}_\mathtt{T}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathbf{y} \cdot (\mathtt{J}_\mathbf{y}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathbf{y})^{-1} \cdot \mathtt{J}_\mathbf{y}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathtt{T}$, or by solving the linear system for $\mathbf{x}$ which has a cubic complexity in the number of frames m — under the assumption that a dense solver is used. An explicit description of how to calculate the scheme (3.24, 3.25) efficiently (using the notion of 'point tracks') are given in Engels et al. (2006) and Konolige (2010).

3.3.6 Solving Sparse Linear Systems

The off-diagonal blocks of the Hessian $\mathtt{J}_\mathtt{T}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathbf{y}$ and $\mathtt{J}_\mathbf{y}^\top \mathtt{\Lambda}_\mathbf{z} \mathtt{J}_\mathtt{T}$ and thus the linear system (3.24) are dense if and only if all points are visible in all frames. In this case $p = m \cdot n$, and the overall computational complexity is $\mathcal{O}(\max(m^3, nm^2))$. Otherwise we do have a second order sparseness structure: Not all points are visible in all frames. Compared to the first order sparseness, which we can exploit using the Schur complement trick, a more general way to exploit the sparseness is needed. There are two successful approaches: Sparse exact solvers and sparse iterative solvers. Within exact solvers, sparse Cholesky (Davis, 2006) is a common solution: It nicely exploits the fact that the normal equation is positive (semi)definite and symmetric, and is therefore more efficient than other approaches (such as sparse LU, or sparse QR). For iterative solvers, *conjugate gradient* is a common choice (Shewchuk, 1994). In a conjugate gradient approach, one typically premultiplies the linear system $\mathtt{A}\mathbf{x} = \mathbf{b}$ to solve with a preconditioner which approximates $\mathtt{A}^{-1}$ in order to start the iteration close to the solution. For BA, a block-diagonal preconditioner has proved to be very effective which can be calculated efficiently too (Jeong et al., 2010).

3.3.7 Robust Least Squares

Let us reformulate the BA energy (3.14) as

$$\chi^2 = \sum_{\mathbf{d}_{i,j}} \left(||\mathbf{d}_{i,j}||_{\mathtt{\Lambda}_{\mathbf{z}_{i,j}}} \right)^2 \quad \text{with} \quad ||\mathbf{d}_{i,j}||_{\mathtt{\Lambda}_{\mathbf{z}_{i,j}}} := \sqrt{(\mathbf{d}_{i,j}(\mathtt{T}_i, \mathbf{y}_j))^\top \mathtt{\Lambda}_{\mathbf{z}_{i,j}} \mathbf{d}_{i,j}(\mathtt{T}_i, \mathbf{y}_j)} \tag{3.26}$$

to emphasize that the error magnitudes have a quadratic influence on χ^2. A single outlier among the measurements would have major negative impact on the estimate, since the quadratic influence of the large error would dominate the cost term. In order to be more outlier-robust, we can replace the quadratic error function $(\cdot)^2$ by a *robust kernel* $\rho(\cdot)$ which weights large errors less. Here, we choose the Huber kernel,

$$\rho_H = \begin{cases} x^2 & \text{if } |x| < b \\ 2b|x| - b^2 & \text{else} \end{cases}, \tag{3.27}$$

which is quadratic for small $|x|$ but linear for large $|x|$. Compared to other, even more robust cost functions, the Huber kernel has the advantage that it is still convex and thus does not introduce new local minima (Hartley & Zisserman, 2004, pp.616). In practice, we do not need to modify equation (3.14). Instead, the following scheme is applied. First the error $\mathbf{d}_{i,j}$ is computed as usual. Then, $\mathbf{d}_{i,j}$ is replaced by a weighted version $w_{i,j}\mathbf{d}_{i,j}$ such that

$$(w_{i,j}\mathbf{d}_{i,j})^\top \Lambda_{\mathbf{z}_{i,j}} (w_{i,j}\mathbf{d}_{i,j}) = \rho\left(\sqrt{\mathbf{d}_{i,j}^\top \Lambda_{\mathbf{z}_{i,j}} \mathbf{d}_{i,j}}\right) \tag{3.28}$$

For the Huber kernel ρ_H these weights are

$$w_{i,j} = \frac{\sqrt{\rho_H(||\mathbf{d}_{i,j}||_{\Lambda_{\mathbf{z}_{i,j}}})}}{||\mathbf{d}_{i,j}||_{\Lambda_{\mathbf{z}_{i,j}}}} \quad \text{with} \quad ||\mathbf{d}_{i,j}||_{\Lambda_{\mathbf{z}_{i,j}}} := \sqrt{\mathbf{d}_{i,j}^\top \Lambda_{\mathbf{z}_{i,j}} \mathbf{d}_{i,j}} \,. \tag{3.29}$$

3.3.8 The g^{2}o Software Package

At this point, we take the opportunity to introduce **g^{2}o**, a graph optimisation library by Kümmerle, Grisetti, Strasdat, Konolige and Burgard (2011a)[3]. On one hand, **g^{2}o** is a very universal software package for least square optimisation. It can solve least square problems of the most general form (2.27), and therefore solves minimisation problems which can be represented by factor graphs. Especially, it can not only deal with the binary constraints which occur in BA, but has also support for unary constraints (e.g. Lovegrove et al., 2011, for fusing GPS measurements) and higher-order constraints (e.g. Kümmerle et al., 2011b, used for sensor calibration). Furthermore in **g^{2}o**, robust kernels can be activated and configured for individual graph constraints. On the other hand, **g^{2}o** is very efficient and does

[3] http://openslam.org/g2o. The implementation is mainly thanks to the first two authors Rainer Kümmerle and Giorgio Grisetti.

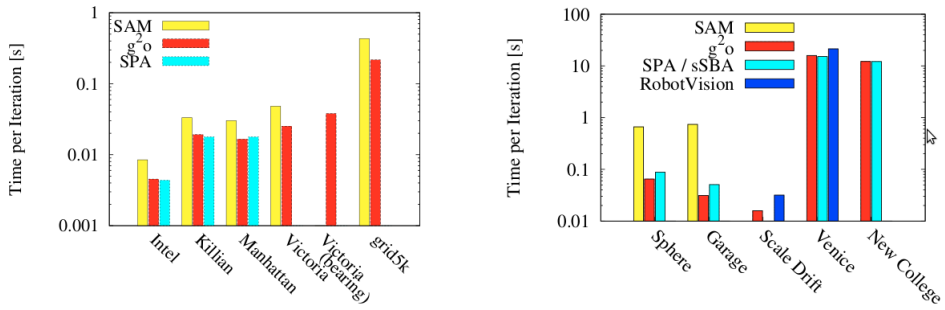


Figure 3.5: Performance comparison of $\mathbf{g^2o}$ with other more specialized optimisation packages (SAM (Dellaert & Kaess, 2006)1, SPA (Konolige et al., 2010), sSBA (Konolige, 2010), RobotVision (Strasdat et al., 2010b)) on different problems/data sets. For more details please refer to Kümmerle et al. (2011a).

offer all the heuristics described above to exploit sparsity. Thus, it supports the Schur complement trick, and it offers various sparse solvers such as sparse Cholesky (Davis, 2006), and block-diagonal preconditioned conjugate gradient. Despite its generality, it can therefore compete with other more application-specific optimization/BA packages. A performance comparison is shown in Figure 3.5. Thus, $\mathbf{g^2o}$ is used throughout this thesis for various least square optimization tasks which require high-end performance.

3.4 Visual Front-end

3.4.1 Bottom-up Matching versus Top-Down Guided Search

Three dimensional objects in the world appear as intensity measurements in camera images. In order to infer the scene geometry, one typically tries to detect primitives in the images. By far the most popular primitives are point features (such as corners, or blobs), even though other primitives such as line-based features are occasionally used too (Smith et al., 2006; Eade & Drummond, 2009). Given a set of camera frames $I_1, I_2, ..., I_m$, we would like to associate two dimensional point measurements $\langle \mathbf{z}_j^{[1]}, \mathbf{z}_j^{[2]}, ..., \mathbf{z}_j^{[m]} \rangle$ among them. Ideally, each such m-tuple of two dimensional points is associated with a single three dimensional point in the world. In this case, each point $\mathbf{z}_j^{[k]}$ in such an m-tuple corresponds to the reprojections of the three dimensional point $\mathbf{y}_j$ in frame I_k. For this problem of feature matching, there are two

canonical approaches: Bottom-up and top-down matching.

In bottom-up approaches, one usually tackles feature tracking in terms of matching correspondences between a pair of images. Thus one associates feature points in one image with feature points in the other image using appearance information only. The challenge is to find features in the image which have a unique appearance and can be easily redetected in other images taken from other view points. To be robust to view-point changes, rotational and scale invariant features such as SIFT (Lowe, 2004) or SURF (Bay et al., 2006) are very popular. Given a set of candidate matches $\tilde{\mathcal{Z}} := \{(\mathbf{z}_1, \tilde{\mathbf{z}}_1), ..., (\mathbf{z}_j, \tilde{\mathbf{z}}_j)\}$, we must look for a geometric constraint between the two images which agrees with the candidate matches. This constraint is described by the essential matrix $\mathtt{E}$ such that $\mathbf{z}_j^\top \cdot \mathtt{E} \cdot \tilde{\mathbf{z}}_j = \mathbf{0}$ (see Section 3.2.1). In order to select a subset of inliers from all candidates $\tilde{\mathcal{Z}}$ which agrees with this *epipolar constraint*, a robust estimation process is required. By far the most popular approaches are based on Random Sample Consensus (RanSaC) and its variants. Here, a random subset is drawn from all candidates $\tilde{\mathcal{Z}}$ and $\mathtt{E}$ is calculated using a closed-form approach — such as Nistér's five point method (2004) or the eight point algorithm (Hartley, 1995). Afterwards, only those pairs in $\tilde{\mathcal{Z}}$ which agree with $\mathtt{E}$ are labelled as inliers. Once an inlier set is found, the initial guess $\mathtt{E}$ can be improved upon using least squares optimisation.

Top-down tracking is model based and therefore a recursive method. The main idea is to guide the feature tracking using our geometric model which was estimated from the previous frames $I_1, ..., I_{t-1}$. First, let us assume that the joint state of the camera pose and the point map is described by a multivariate Gaussian, e.g. when using EKF-based SLAM. Also we assume we have some kind of motion prior which allows us to predict the distribution over the camera pose $\langle \mathtt{T}_t, \Sigma_{\mathtt{T}_t} \rangle$ before any measurements from $\mathtt{I}_t$ are integrated. Then, we can calculate the distribution in the image space – the mean $\hat{\mathbf{z}}(\mathtt{T}_t \cdot \mathbf{y})$ and the innovation covariance

$$\mathtt{S} = \begin{pmatrix} \left(\frac{\partial}{\partial_{\mathbf{x}}} \hat{\mathbf{z}}(\exp(\widehat{\boldsymbol{\epsilon}})\mathtt{T}_t \cdot \mathbf{y}_j)\right)^\top_{\boldsymbol{\epsilon}=0} \\ \frac{\partial}{\partial_{\mathbf{y}}} \hat{\mathbf{z}}(\mathtt{T}_t \cdot \mathbf{y}_j)^\top \end{pmatrix}^\top \begin{bmatrix} \Sigma_{\mathtt{T}_t} & \Sigma_{\mathtt{T}_t,\mathbf{y}} \\ \Sigma^\top_{\mathtt{T}_t,\mathbf{y}} & \Sigma_{\mathbf{y}} \end{bmatrix} \begin{pmatrix} \frac{\partial}{\partial_{\mathbf{x}}} \hat{\mathbf{z}}(\exp(\widehat{\boldsymbol{\epsilon}})\mathtt{T}_t \cdot \mathbf{y}_j)\Big|_{\boldsymbol{\epsilon}=0} \\ \frac{\partial}{\partial_{\mathbf{y}}} \hat{\mathbf{z}}(\mathtt{T}_t \cdot \mathbf{y}_j) \end{pmatrix} + \Sigma_{\mathbf{z}} \tag{3.30}$$

where the point $\mathbf{y}_j$ is expected to be seen in the image. Thus, $\mu_{\hat{\mathbf{z}}}$ and $\mathtt{S}$ define an elliptical region in the image which can be used for to actively search for features at their expected locations as described by Davison (2005).

To summarize, bottom-up matching has the advantage that it does not require any motion prior and can therefore detect matches between frames with arbitrary camera configuration. However, it is computationally expensive since it does not only requires to touch every single pixel in the image at least once, but also depends on a high number of RanSaC iterations to run robustly. On the other hand, top-down matching is model-based, and more efficient since feature matching can be restricted to small elliptical search regions. However, it usually requires an uncertainty estimate for the point maps as well as a narrow motion prior.

3.4.2 Optical Flow-guided Search

We suggest a framework that combines bottom up-tracking and top-down search techniques. The idea is to perform first a per pixel association between images using brightness information, called *optical flow*. Afterwards, we continue with guided feature matching using the current map prediction and the optical flow estimate.

In optical flow, we try to find for each image coordinate $\mathbf{z} = (u, v)$ a flow vector $\mathbf{q_z} = (q_\mathbf{z}^{(1)}, q_\mathbf{z}^{(2)})$ such that $|I_t(\mathbf{z} + \mathbf{q_z}) - I_{t-1}(\mathbf{z})|$ is minimal. Given the images have a resolution of $W \cdot H$, it follows that optical flow is a high-dimensional problem with $2(W \cdot H)$ unknowns. The problem is clearly under-constrained, since the residual error has only $W \cdot H$ dimensions. One way out is to introduce soft constraints which enforce that flow vectors close by should be of similar length and orientation. This kind of soft constraint which adds a penalising cost to the energy is called a *regulariser*. One often enforces that the spatial gradient of the flow field $\nabla \mathbf{q}$ stays small; an idea which dates back to the seminal work of Horn & Schunck (1981). One possible optical flow formulation using such a regulariser is:

$$\min_{\mathbf{q}} \left\{ ||\nabla \mathbf{q}||_1 + \lambda \sum_{u,v} \left| I_t \left(u + q_{u,v}^{(1)}, v + q_{u,v}^{(2)} \right) - I_{t-1}(u, v) \right| \right\} \quad (3.31)$$

$$\text{with} \quad ||\nabla \mathbf{q}||_1 := \sum_{u,v} \sqrt{\left(\frac{\partial q_{u,v}^{(1)}}{\partial u} \right)^2 + \left(\frac{\partial q_{u,v}^{(1)}}{\partial u} \right)^2 + \left(\frac{\partial q_{u,v}^{(2)})}{\partial v} \right)^2 + \left(\frac{\partial q_{u,v}^{(2)}}{\partial v} \right)^2} \,. \quad (3.32)$$

Here, λ is the essential parameter which weights the influence between the regulariser and the data term. Instead of using a quadratic error function, the 1-norm is used for the regulariser and the data term so that the minimisation is robust to outliers, e.g. due to occlusions, specular highlights etc. Note that the energy (3.31) does not

only depend on $\mathbf{q}$ but also on the gradient $\nabla\mathbf{q}$ which can be seen as a functional (= higher-order function) over $\mathbf{q}$. The theory of minimising funtionals is called *calculus of variation*, thus this visual tracking approach is denoted as *variational optical flow*. Variational optical flow is highly parallelisable, since the regulariser and the data term can be computed for each pixel independently, and can therefore be efficiently implemented on a modern GPU. A particularly efficient solution for this minimisation problem is given by the primal dual algorithm of Chambolle & Pock (2011). In order to increase the basin of convergence, typically an image pyramid approach is employed (Adelson et al., 1984). We used the 'FlowLib' implementation which is available online.[4]

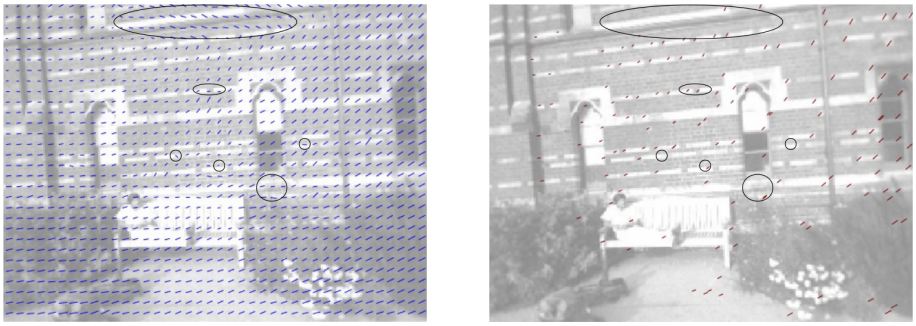
Even though variational optical flow can lead to high quality results, there is no mechanism which enforces that the flow field is consistent with the camera motion. Especially if there is repetitive structure in the scene, it can easily happen that a flow field is generated which is partially wrong as illustrated in Figure 3.6(a). Therefore, we suggested the following approach: First we calculate an optical flow field between the previous and the new frame. Afterwards, we project all visible points in our model $\mathbf{y}_j$ into the previous frame $\mathtt{I}_{t-1}$ and push these coordinates $\hat{\mathbf{z}}(\mathtt{T}_{t-1}\cdot\mathbf{y}_j)$ through the flow field from $\mathtt{I}_{t-1}$ to $\mathtt{I}_t$. The resulting predictions $\hat{\mathbf{z}}(\mathtt{T}_{t-1}\cdot\mathbf{y}_j)$ $+ \mathbf{q}_{\hat{\mathbf{z}}(\mathtt{T}_{t-1}\cdot\mathbf{y}_j)}$ in the new image are used to estimate the new pose $\mathtt{T}_t$ by minimizing the following energy:

$$\chi^2(\mathtt{T}_t) = \sum_i \rho_H \left(\hat{\mathbf{z}}(\mathtt{T}_{t-1}\cdot\mathbf{y}_j) + \mathbf{q}_{\hat{\mathbf{z}}(\mathtt{T}_{t-1}\cdot\mathbf{y}_j)} - \hat{\mathbf{z}}(\mathtt{T}_t\cdot\mathbf{y}_j) \right)^2 . \tag{3.33}$$

We employ a robust kernel ρ_H so that the pose estimation of $\mathtt{T}_t$ is robustly fitted and flow vectors which violate the epipolar constraint only have a minor impact. The initial guess for $\mathtt{T}_t$ is simply set to $\mathtt{T}_{t-1}$, thus no motion model is required. This approach only fails if a large portion of the flow field is wrong (e.g. because the motion is too large) or the rigid scene assumption is heavily violated.

Once we have a good pose estimate $\mathtt{T}_t$, we can perform a guided search to verify feature matches. As described in equation (3.30), the uncertainty of the feature location and thus the size and shape of the search region depends on the uncertainty of map point $\mathbf{y} \in \mathcal{Y}$ and the camera pose $\mathtt{T}_{t+1}$ as well as the measurement uncertainty $\Sigma_{\mathbf{z}}$. Given that all points in $\mathcal{Y}$ are already optimised using keyframe bundle adjustment, and that the pose $\mathtt{T}_{t+1}$ is well estimated using the robust optical

[4] http://www.gpu4vision.org



(a) optical flow field (b) robustly fitted feature tracks

Figure 3.6: Optical flow field (a) containing regions of outliers; six such regions are highlighted above. Feature tracks (b) are robustly fitted to the flow field.

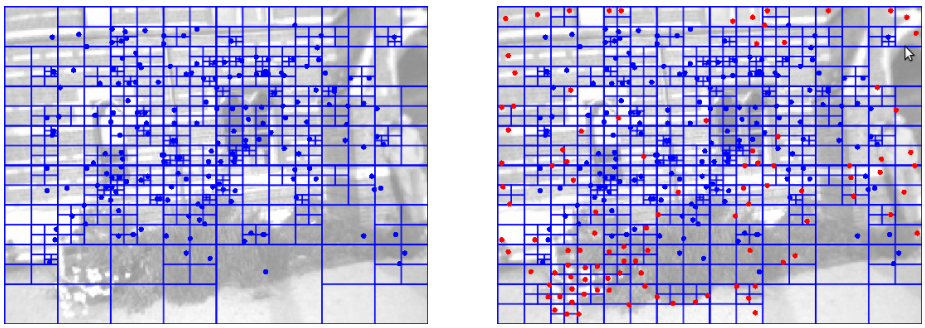
flow constraint, it is sufficient to consider merely a small circular search region of a few pixels for feature matching. The search within this region is based on template matching using normalised sum of squares. Using the pose estimate $\mathtt{T}_{t+1}$, the target templates can be warped accordingly. As in PTAM (Klein & Murray, 2007), we do not apply the matching at every single pixel in the search region, but only there where FAST features (Rosten & Drummond, 2006) were detected in order to speed-up the search even further. As a result, we receive a set of 2D-3D matches between image locations $\mathbf{z}_k$ and feature points $\mathbf{y}_k$. Finally, we can optimize the pose $\mathtt{T}_{t+1}$ even further using *motion-only BA*. Thus, we minimize

$$\chi^2(\mathtt{T}_t) = \sum_k (\mathbf{z}_k - \hat{\mathbf{z}}_k(\mathtt{T}_t \cdot \mathbf{y}_j))^2 \tag{3.34}$$

wrt. $\mathtt{T}_t$ using Levenberg-Marquardt. Resulting feature tracks are shown in Figure 3.6(b).

3.4.3 Feature Selection using a Quadtree

An image recorded using a digital camera typically consists of tens of thousands to millions of pixels. For image processing tasks such as feature initialisation for visual SLAM, it is crucial to select the relevant parts of the image in order to bound the computational cost. We can perform a preselection using keypoint detection mechanism such as FAST. While such keypoints usually have some kind of tunable thresholds which allow coarse variation of the number of features per image or image region (e.g. by filtering out extrema in the intensity image with a low strength),



(a) Old map points (b) New features added

Figure 3.7: Quadtree used for feature initialisation. New (red) features are only added in regions with low feature density.

a more fine-grained control is usually required. For instance, it is a good idea to initialize new features in those image regions where the feature density is low. In order to implement such a strategy efficiently, it is a good idea to store image features in an appropriate data structure which embodies the spatial relations in the image. One such representation is a (region) quadtree (Finkel & Bentley, 1976). Here, each tree node represents an image region, so that the root represents the whole image. Each node has either zero or four children. A parent node partitions its region equally among its children: top left quarter, top right quarter, bottom left quarter and bottom right quarter. The content, 2D image points, is only stored at the leaves of the tree. On the other hand, there are leaves which do not hold any points. This becomes obvious if we consider a simple example such as a quadtree which is storing two points. By definition it must have four leaves. It is often convenient to define a minimum region size in order to control the maximum possible point density and also the maximum tree height. We follow the approach of Mei et al. (2009) who suggested to storing image features in this data structure to promote a uniform feature distribution. Especially, one can base the decision whether to add a candidate feature $\mathbf{z}$ to the image on how many features are already in the bounding box around $\mathbf{z}$ as illustrated in Figure 3.7.

However, only being selective during feature initialization is not sufficient; one has to be careful during feature tracking too. Even if one is selective in the feature initialisation process, situations can easily arise where thousands of points are visible in the current image, while the prepossessing budget only allows one to measure a small subset of them. Thus, it can be beneficial to have a mechanism for selecting

Algorithm 1 BFS quadtree traversal, returns next node

```
global queue : Queue<pointer<Node> >

while queue.size()>0 do
  pointer<Node> node := queue.pop()
  if node->children.size()=0 then      //node is leaf
    if node->constains_point then
       return node->point2d
    end
  else                                 //node has 4 children
    queue.append(node->children[0])
    queue.append(node->children[1])
    queue.append(node->children[2])
    queue.append(node->children[3])
  end
end
throw Exception("Reached end")
```

e.g. a hundred representative features out of thousands of unevenly distributed keypoints. At first glance, the following heuristic seems to provide a uniformly distributed feature selection: We store all candidates in a quadtree, and then perform a breadth first search (BFS) traversal as described in Algorithm 1. Following this approach, the image regions and sub-regions are indeed traversed in an approximated uniformly manner. However, this strategy does not lead to the desired result since keypoints are only stored in the leaves of the quadtree. In particular, BFS traversal leads to a biased selection where leaves of low levels, and thus features which lie in low density regions, are selected first.

Instead, we suggest the following advanced traversal strategy which is essentially a combination of BFS and depth first search (DFS). First, a node is selected using BFS. If it is a leaf, there are two options: Either the node carries a point in which case we return it. Otherwise, we continue with BFS. However, if the node is not a leaf, we still wish to return a keypoint in its region. Thus, we perform DFS to find one of its leaves which carries a point (and which was not returned before). The details of the algorithm, that allows for a equi-distributed feature traversal, are listed in Algorithm 2. In order to diminish further bias, it is important to traverse/select the children of a node in non-deterministic way. The difference between BFS traversal and our advanced traversal strategy are compared in Figure 3.8. While BFS has

Algorithm 2 Equi-distributed quadtree traversal, returns next node

```
global  map : Map<Integer, List<pointer<Node> > >

while map.size()>0 do
  //do BFS with random sibling selection
  level := map.getSmallestKey()
  if map[level].size()=0 then
    map.eraseElement(level)
    continue
  end
  pointer<Node> node := map[level].popRandomElement()
  if node->children.size() = 0 then
    //node is a leaf
    if node->contains_point and node->visited=false then
      //only return leaf, if it contains a point and was not returned before
      node->visited := true
    end
  else
    if not map.hasKey(level+1) then
      map.insert(level+1,[])
    end
    map[level+1].append(node->children[0])
    map[level+1].append(node->children[1])
    map[level+1].append(node->children[2])
    map[level+1].append(node->children[3])

    //perform DFS with random child traversal to return single entry
    define stack : Stack
    stack.addChildrenInRandomOrder(node->children)
    while stack.size()>0 do
      pointer<Node> dfs_node := stack.pop()
      if dfs_node->children.size()=0 then
        if dfs_node->contains_point
           and dfs_node->visited=false
        then
          dfs_node->visited := true
          return dfs_node->point2d
        end
      else
        stack.addChildrenInRandomOrder(dfs_node->children)
      end
    end
  end
end
throw Exception("Reached end")
```

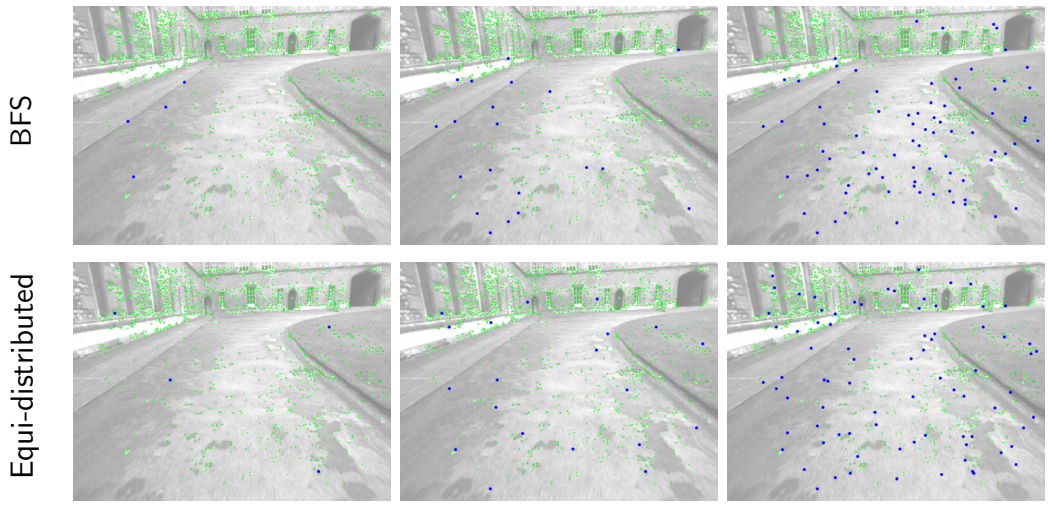
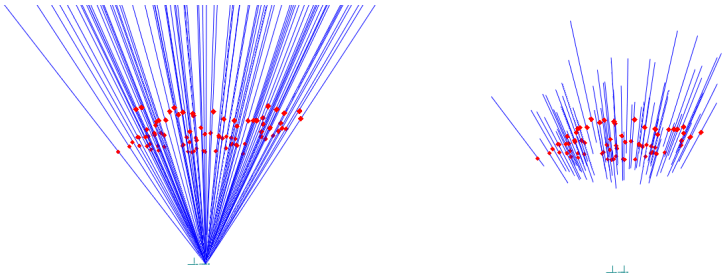


Figure 3.8: Comparison of BFS and equi-distributed quadtree traversal to select 4, 20, and 80 representative feature from a set of over thousand keypoints.



(a) Feature initialisation (b) After a single update

Figure 3.9: Illustration of the feature initialisation process. First features are initialised as inverse depth points with infinite uncertainty in depth (a). A single update leads to a significant reduction of the depth uncertainty (b). The inverse depth features are plotted using a 99.7 percent confidence interval.

a strong preference to select features from low density areas, our advanced strategy results in approximately uniform feature distribution.

3.4.4 Feature Initialisation

In monocular SLAM approaches using filtering such as MonoSLAM, no special treatment for feature initialisation is needed if an inverse depth representation (Montiel

et al., 2006) is used. New features are jointly estimated together with all other parameters, but at a cost of $O(n^3)$ where n is the number of features visible. In keyframe-based SLAM approaches a dedicated feature intialisation scheme is required. Finding feature matches is difficult in a top down manner, since such a partially initialised feature with unknown depth can lie anywhere on the epipolar line of the subsequent keyframe. Typically, a strong depth prior is enforced in order to restrict the search and therefore minimize matching ambiguities (Klein & Murray, 2007). Once feature matches are established, features can be triangulated between keyframes. We suggest a feature initialisation method based on a set of three dimensional information filters which can estimate the position of arbitrarily distant features. A similar method was briefly described by Klein & Murray (2009).

Ultimately, we would like to update a set of partially initialised 3D points $\mathbf{y}_{new:j}$ given the current camera pose $\mathtt{T}_t$. If $\mathtt{T}_t$ is known, the features $\mathbf{y}_{new:n}$ become independent:

$$p(\mathbf{y}_{new:1}, ..., \mathbf{y}_{new:j}, ..|\mathtt{T}_i) = p(\mathbf{y}_{new:1}) \cdots p(\mathbf{y}_{new:n}) \ . \tag{3.35}$$

Since the current camera $\mathtt{T}_t$ is well-optimised wrt. the set of map points $\mathcal{Y}$, the independence assumption is approximately true. Thus, our method employs a set of information filters. Each filter estimates the position of a single landmark given the current pose estimate. In this sense, our approach has some similarities to FastSLAM (Montemerlo & Thrun, 2003). The difference to FastSLAM is that the partially initialised features $\mathbf{y}_{new:j}$ are not used for state estimation immediately. New features are only used for pose estimation after they are jointly bundle adjusted and added to the map $\mathcal{Y}$.

The design of a single filter is inspired by Eade's filtering framework (2008). Features are represented using an inverse depth parametrisation $\boldsymbol{\psi}$ wrt. the origin keyframe $\mathtt{T}_{cw}$ in which they were seen first. Here, ψ_3 represents inverse depth, whereas (ψ_1, ψ_2) are normalised pixel coordinates. The anchored inverse depth point $\boldsymbol{\psi}$ can be mapped to a Euclidean point $\mathbf{y} = \mathtt{T}_{cw}^{-1} \cdot (\frac{\psi_1}{\psi_3}, \frac{\psi_2}{\psi_3}, \frac{1}{\psi_3})^\top$ in this global coordinate frame. The uncertainty of an inverse depth feature is represented using the information matrix $\boldsymbol{\Lambda}_{\boldsymbol{\psi}}$. In each keyframe, we initialise new features. Appropriate locations are determined using a quadtree (as described above). Given the feature location $\mathbf{z} = (u, v)^\top$, we set $\boldsymbol{\psi} = (\frac{u-p_1}{f}, \frac{v-p_2}{f}, q)$ with $q \in \mathbb{R}^+$. The uncertainty $\boldsymbol{\Lambda}_{\boldsymbol{\psi}}^{(0)}$ is set to $\mathrm{diag}(\frac{f^2}{\sigma_{\mathbf{z}}^2}, \frac{f^2}{\sigma_{\mathbf{z}}^2}, 0)$. Note that initially there is an infinite uncertainty along the feature depth, so we do not enforce any depth prior no matter which start value we

assign for q.

We employ a Gauss-Newton filter scheme (Section 2.3.4) to minimise

$$\chi^2(\boldsymbol{\psi}) = (\boldsymbol{\psi} - \hat{\boldsymbol{\psi}})^\top \Lambda_{\boldsymbol{\psi}} (\boldsymbol{\psi} - \hat{\boldsymbol{\psi}}) + (\mathbf{z} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}))^\top \Lambda_{\mathbf{z}} (\mathbf{z} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y})) \tag{3.36}$$

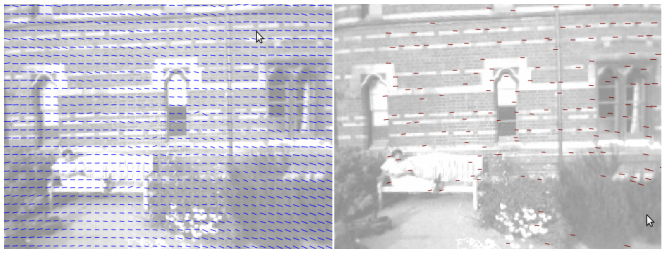
wrt. $\boldsymbol{\psi}$ using Levenberg Marquardt. The first term in $\chi^2(\boldsymbol{\psi})$ ensures that the optimisation of $\boldsymbol{\psi}$ is based on its prior distribution $\langle \hat{\boldsymbol{\psi}}, \Lambda_{\boldsymbol{\psi}} \rangle$. The second term takes care that the projection error with respect to the current frame is reduced. In other words, the point $\boldsymbol{\psi}$ is moved along its uncertain depth rather than via its certain u, v coordinate in order to reduce the projection error in the current image. We also update the uncertainty using uncertainty propagation (see Figure 3.9):

$$\Lambda_{\boldsymbol{\psi}}^{(k+1)} = \Lambda_{\boldsymbol{\psi}}^{(k)} + \left(\frac{\partial \Delta \mathbf{z}}{\partial \boldsymbol{\psi}} \right)^\top \Lambda_{\mathbf{z}}^{(k)} \left(\frac{\partial \Delta \mathbf{z}}{\partial \boldsymbol{\psi}} \right) . \tag{3.37}$$

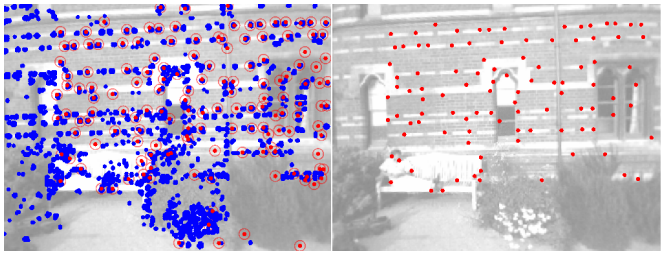
3.5 Qualitative Experiment

We performed the evaluation of our monocular exploration system using the Keble College data set of Clemente et al. (2007) — a sequence where a hand-held sideways-facing camera completes a circuit around a large outdoor square. Images were captured using a low cost IEEE Unibrain camera with resolution 320×240, and using a lens with 80 degree horizontal field of view. Our framework performed at near real-time at approximately 12 FPS. The computation was performed on a desktop computer with an Intel Core 2 Duo processor and an NVIDIA 8800 GT GPU which were used for dense optical flow computation.

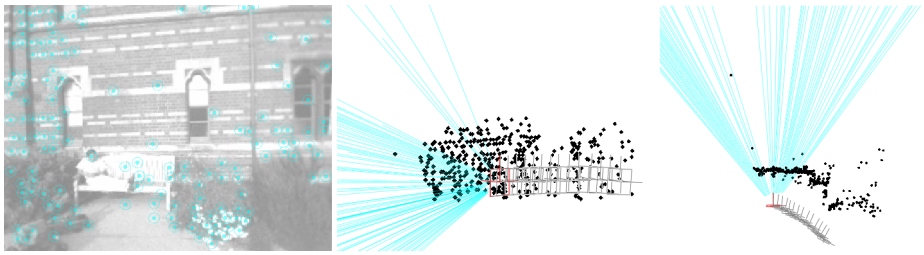
To bootstrap the joint structure and motion problem, we performed optical flow based feature tracking and the classic 8-point algorithm (Hartley & Zisserman, 2004) in the conjunction with RanSaC (Fischler & Bolles, 1981) to find a robust structure and motion estimate. A qualitative evaluation of monocular exploration is presented in Figure 3.10. First, we illustrate the feature and pose tracking pipeline. In (a), variational optical flow is shown; feature tracks and therefore the common incremental motion is fitted to the flow field (b), which leads to an initial pose estimate (c). Now, as illustrated in (d), guided search is performed in elliptical search regions (red), whereas patch matching is only performed where FAST keypoints (blue) are found. Successful matches and the refined pose are shown in (e) and (f) respectively. Then, (g) illustrates the initialisation of inverse depth features, while the feature and



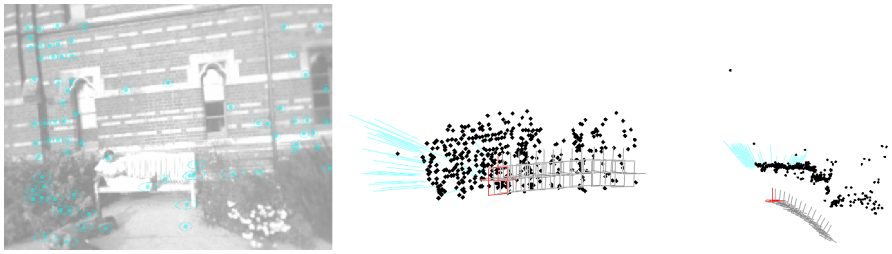
(a) Optical flow (b) Fitted feature tracks (c) Initial pose estimate



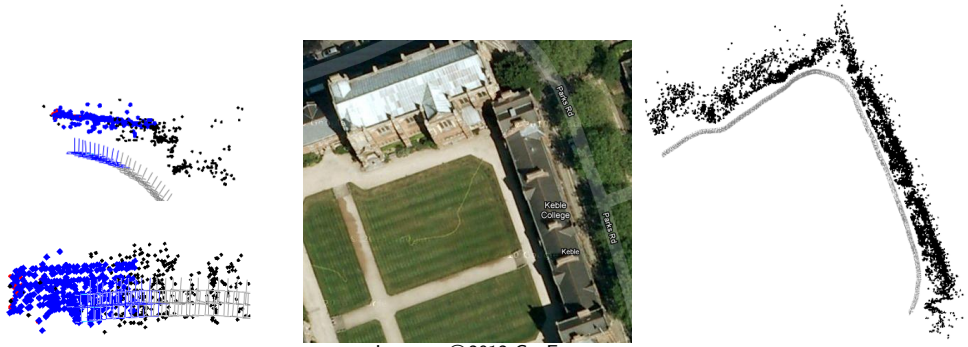
(d) Guided matching (e) Found features (f) Pose refinement



(g) Initialisation of inverse depth points



(h) Tracking and update of inverse depth points



(i) Keyframe sliding window (j) Imagery ©2012 GeoEye, Getmapping plc, Infoterra Ltd & Bluesky, The GeoInformation Group, Map data ©2012 Google (k) Reconstruction

Figure 3.10: Keble College experiment

depth estimates after several updates are shown in (h). Keyframe sliding window optimisation is shown in (i). In particular, we employ bundle adjustment using the Schur complement trick and sparse Cholesky to exploit the sparsity of the Hessian as well as a Huber kernel, so that the optimisation is robust to the small fraction of spurious matches. Finally, (j) shows an aerial image of the Keble college campus, while (k) shows the 3D reconstruction of the first half of the dataset. Results on the full data set, are shown in Chapter 5 where we complete the monocular SLAM framework by discussing the problem of scale drift and by presenting a framework for loop closure correction. Furthermore, a quantitative evaluation of the accuracy and cost of bundle adjustment is given in the following chapter.

3.6 Summary

In this chapter, we presented a framework for visual SLAM, with the emphasis on monocular exploration. This chapter mainly served as an introduction to visual SLAM on a technical level. In particular, we introduced various related concepts, such as the pinhole camera model, optimization back-end versus visual front-end, gauge freedom in optimization and the monocular scale ambiguity as well as techniques, e.g. efficient bundle adjustment, the Schur complement trick, guided feature tracking, and robust least squares, which are essential for the remainder of this work. However, we also presented some novelties such as optical flow guided feature and pose tracking as well as equi-distributed quadtree traversal. In addition, we introduced the $\mathbf{g^2o}$ optimization framework of Kümmerle et al. (2011a) which will be used throughout this thesis.

3.7 Bibliographic Remarks

The origin of on bundle adjustment dates back to the work of Brown (1958) in the context of photogrammetry. Note that he already applied the Schur complement trick to exploit the first order sparseness structure, so that the underlying linear system only depends on the number of frames. A vast quantity of work on bundle adjustment was published in the last decades. An excellent and comprehensive survey on bundle adjustment was presented by Triggs et al. (1999), which can be still

seen as the standard reference today. In 2006, Engels et al. highlighted the usefulness of bundle adjustment for real-time camera tracking. Especially, they showed how the organisation of the data in *point tracks* can lead to efficient implementation of the outer product. According to Triggs et al. (1999), Gyer & Brown (1967) were the first to exploit the second order sparsity, i.e. that not all points are visible in all frames, using recursive partitioning. Recently, it caught on to exploit the second order sparsity using either *sparse Cholesky* (Davis, 2006), e.g. Agarwal et al. (2009), Lourakis & Argyros (2009), Strasdat et al. (2010b), Konolige (2010), or preconditioned conjugate gradient as in Byrod & Astrom (2010) and Jeong et al. (2010).

Feature matching is very commonly done with a bottom-up approach, where in a first abstraction step some keypoints are extracted from the target images. While initial structure from motion approaches applied corner features such as Harris & Stephens (1988) or Shi & Tomasi (1994), rotation and scale invariant blob features are very common nowadays. Most prominently is the Scale Invariant Feature Transform (SIFT) of Lowe (1999, 2004), which consists of a difference of Gaussian detector and a histogram of gradient orientations descriptor. In 2006, Bay et al. presented Speeded-Up Robust Features (SURF) which employs integral images to compute determinant of Hessian keypoints and distribution-based descriptors efficiently. In 2011, even more efficient blob features based on Calonder et al.'s Binary Robust Independent Elementary Features (BRIEF) were presented by Rublee et al. (2011) and Leutenegger et al. (2011). For bottom-up approaches, RanSaC (Random Sample Concensus, Fischler & Bolles, 1981) and its variants such as MSaC and MLESaC (Torr & Zisserman, 2000) are the de facto standard to separate inliers from outliers. Underlying models are typically based on the epipolar constraints and can be solved in closed form using either $\{7, 8\}$-point approaches (Hartley & Zisserman, 2004) or, if the camera intrinsics are known, using Nistér's 5-point method (2004).

In contrast to bottom-up techniques, top-down tracking is model-based. Davison (2003, 2005) incorporated prior knowledge in terms of a Gaussian map estimate as well as a velocity-based motion model to restrict the search window for feature tracking. In order to test whether those matched features mutually agree, Neira & Tardós's 'joint compatibility test' (2001) can be used. Alternatively, Chli & Davison (2009) showed that guided search can be extended to an advanced active matching algorithm where the predictions are updated after each match so that the

search regions shrink continuously. Here, multi-hypothesis are maintained in order to deal with mismatches. In PTAM, Klein & Murray (2007) used top-down guided tracking with constant circular search region in a pyramidal approach. Matching is only performed for pixels where FAST keypoints (Rosten & Drummond, 2006) were detected.

Optical flow is commonly used for feature tracking. In visual SLAM, the KLT approach (Lucas & Kanade, 1981; Tomasi & Kanade, 1991) is sometimes applied which performs local searches for a sparse set of features (Rybski et al., 2003; Klippenstein & Zhang, 2007; Lim et al., 2011). Our guided feature tracking using variational optical flow is related to the work of Wedel et al. (2008). In contrast to our approach, where we first estimate the flow field and then fit robustly the motion, Wedel et al. estimate a flow field which is coherent with the epipolar constraint by adding a corresponding cost term to the energy. This approach is beautiful since it always estimates a motion-coherent flow field; but it is not straight forward when the essential matrix is not given a priori. Despite the fact that it depends on a slow alternation approach in this case, it is less clear what a good initial guess for the optimisation would be since the space of essential matrices has a singularity around zero (see Figure 3.2).

In the original MonoSLAM, Davison (2003) used a set of particles to represent the unknown depth of newly initialised features, while Lemaire et al. (2005) and Solà et al. (2005) used mixture of Gaussian approaches. The inverse depth formulation of Montiel et al. (2006), which allows representation of even partially initialized features well using a Gaussian distribution, first made it possible to deal with monocular features in a unified manner. No special treatment for partially initialised features with unknown depth is required. Though, this unified approach is only valid if the map is represented jointly using a multi-variate Gaussian distribution. In keyframe-based optimisation approaches, special treatment for feature initialisation is still required. Klein & Murray (2007) used a depth prior in order to restrict the search on the epipolar line. The feature initialisation method introduced in this chapter, that was first described in Strasdat et al. (2010b), was developed independently from Klein & Murray (2009) which was published half a year earlier. On a technical level, our approach differs from theirs' by using of three dimensional feature representation instead of filtering only depth (= one dimension).

Quadtrees were introduce by Finkel & Bentley (1976). They showed that insertion

and search can be performed efficiently. Quadtrees, and related data structures using recursive partitioning, are commonly used for applications in image processing and computer graphics (Samet, 1984). Mei et al. (2009, 2010a) suggested to use a quadtree for feature tracking in the context of visual SLAM.

In the beginning of the past decade, several promising systems for incremental structure and motion estimation were developed such as Zhang & Shan (2003) and Corke et al. (2004). Nistér et al. (2006) presented a framework for visual odometry using monocular and stereo vision. The approach is based on bottom-up tracking using Harris corners (Harris & Stephens, 1988) and RanSaC. For the monocular approach it is based on the five point method (Nistér, 2003) and for stereo vision it uses the three point method (Haralick et al., 1994). Sliding window bundle adjustment is used for iterative refinement of structure and motion. Nistér et al. (2004) showed a qualitative evaluation for real-time monocular exploration and extensive results for real-time visual odometry using stereo vision over trajectories of hundreds of meters. Mouragnon et al. (2006) presented a framework for monocular exploration which relies on bundle adjustment in a sliding window of keyframes. Using a three point RanSaC approach, features are tracked against the 3D model. The system runs in real-time on 7.5 fps image sequences and produces accurate results. Konolige et al. (2007) presented a real-time approach for stereo visual odometry using sliding window BA, and performed experiments over kilometer-long trajectories. Many other stereo visual odometry frameworks followed such as Kaess et al. (2009), Beall et al. (2010), Geiger et al. (2011) and Alcantarilla et al. (2012). Civera et al.'s monocular framework (2010) makes use of a camera-centric parametrisation and performs inference by means of an EKF. A novel 1-point RanSaC scheme is used to enable robust model-based tracking. Large scale results show accuracies comparable to approaches based on bundle adjustment, while the computational performance is short of real-time at 1 fps (under the absence of wheel odometry).

This chapter is partially based on Strasdat et al. (2010b).

Chapter 4

Visual SLAM: Why Filter?

In which we compare rigorously the relative advantages of Gaussian filters versus keyframe bundle adjustment for real-time visual SLAM.

While the most accurate solution to offline structure from motion problems is undoubtedly to extract as much correspondence information as possible and perform batch optimisation, sequential methods suitable for live video streams must approximate this to fit within fixed computational bounds. Live motion and structure estimation from a single moving video camera has a long history dating back to work such as Harris & Pike (1987), but recent years — through advances in computer processing power as well as algorithms — have seen great progress. Two quite different approaches to real-time visual SLAM have proven successful, but they sparsify the problem in different ways. *Filtering* methods (e.g. Jung & Lacroix, 2003; Davison et al., 2007; Eade & Drummond, 2007; Pietzsch, 2008) marginalise out past poses and summarise the information gained over time with a probability distribution. *Keyframe* methods (e.g. Mouragnon et al., 2006; Klein & Murray, 2007; Pirker et al., 2011) retain the optimisation approach of bundle adjustment (BA), but computationally must select only a small number of past frames to process.

Understanding of the generic character of localisation and reconstruction problems has recently matured significantly. In particular, recently a gap has been bridged between the structure from motion research area in computer vision and the SLAM sub-field of mobile robotics research. The essential character of these two problems,

estimating sensor motion by modelling the previously unknown but static environment, is the same, but the motivation of researchers has historically been different. Structure from motion tackled problems of 3D scene reconstruction from small sets of images, and projective geometry and optimisation have been the prevalent methods of solution. In SLAM, on the other hand, the classic problem is to estimate the motion of a moving robot in real-time as it continuously observes and maps its unknown environment with sensors which may or may not include cameras. Here sequential filtering techniques have been to the fore.

It has taken the full adoption of Bayesian methods for both to be able to be understood with a unified single language and a full cross-over of methodologies to occur. We will discuss several approaches which aim at pulling together the best of both worlds. There remains, however, the fact that in the specific problem of real-time camera tracking, the best systems have been strongly tied to one approach or the other. The question of why, and whether one approach is clearly superior to the other, needs resolving to guide future research in this important application area.

4.1 Filtering versus Bundle Adjustment

Let us recapitulate that the general problem of SLAM can be posed in terms of inference on a graph.[1] We represent the variables involved by the graphical models shown in Figure 4.1(a). The variables of interest are $\mathtt{T}_i$, each a vector of parameters representing a historic pose of the camera, and $\mathbf{y}_j$, each a vector of parameters representing the position of a 3D feature, assumed to be static. These are linked by image feature measurements $\mathbf{z}_{ij}$ — the observation of feature $\mathbf{y}_j$ from pose $\mathtt{T}_i$ — represented by edges in the graph. In real-time SLAM, this network will continuously grow as new pose and measurement variables are added at every time step, and new feature variables will be added whenever new parts of a scene are explored for the first time. Although various parametric and non-parametric inference techniques have been applied to SFM (structure from motion) and SLAM problems such as particle filters (Sim et al., 2005; Eade & Drummond, 2006), or global optimisation based on the L_∞-norm (Hartley & Schaffalitzky, 2004), the most generally success-

[1] The close relation between graphical models and structure of the corresponding least squares problem was discussed in detail by Thrun et al. (2002) and Dellaert & Kaess (2006).

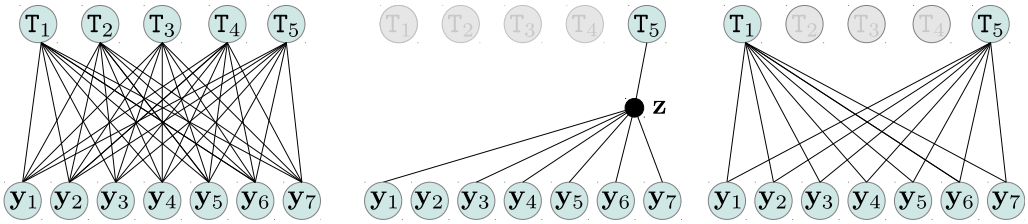


Figure 4.1: (a) illustrates the full SFM problem as a Markov random field, which can be seen as a special factor graph with only binary constraints. Measurements $\mathbf{z}_{i,j}$ (= binary constraints) are represented by an edge between poses $\mathtt{T}_i$ and features $\mathbf{y}_j$. (b) shows sequential filtering in a factor graph. (c) shows the sparsification in keyframe-based optimisation.

ful methods in both filtering and optimisation have assumed Gaussian distributions for measurements and ultimately state-space estimation; equivalently we could say that they are least-squares methods which minimise the reprojection error. Bundle adjustment (BA) in structure from motion, or the Extended Kalman Filter (EKF) and variants in SLAM, all manipulate the same types of matrices representing Gaussian means and covariances. The clear reason is the special status of the Gaussian as the central distribution of probability theory which makes it the most efficient way to represent uncertainty in a wide range of practical inference (Jaynes, 2003). We therefore restrict our analysis to this domain.

A direct application of optimal BA to sequential SLAM would involve finding the full maximum likelihood solution to the graph of Figure 4.1(a) from scratch as it grew at every new time-step. The computational cost would clearly get larger at every frame, and quickly out of hand. In inference suitable for real-time implementation, we therefore face two key possibilities in order to avoid computational explosion. In the *filtering* approach illustrated by Figure 4.1(b), all poses other than the current one are marginalised out after every frame. Features, which may be measured again in the future, are retained. The result is a graph which stays relatively compact; it will not grow arbitrarily with time, and will not grow at all during repeated movement in a restricted area, adding persistent feature variables only when new areas are explored. The downside is that the graph quickly becomes fully interconnected, since every elimination of a past pose variable $\mathtt{T}_{i-1}$ causes fill-in in terms of a high-order constraint $\hat{\mathbf{z}}(\mathtt{T}_i, \mathbf{y}_1, ..., \mathbf{y}_j)$ between all feature variables to which it was joined. A joint distribution over all of these interconnected variables must therefore

be stored and updated. The computational cost of propagating joint distributions scales poorly with the number of variables involved, and this is the main drawback of filtering: in SLAM, the number of features in the map will be severely limited. The standard algorithm for filtering using Gaussian probability distributions is the EKF, where the dense inter-connections between features are manifest in a single joint density over features stored by a mean vector and large covariance matrix.

The other option is to retain BA's *optimisation* approach, solving the graph from scratch time after time as it grows, but to sparsify it by removing all but a small subset of past poses. As discussed in Section 3.3.3, it is sometimes sensible for the retained poses to be in a sliding window of the most recent camera positions, but more generally they are a set of intelligently or heuristically chosen keyframes (see Figure 4.1(c)). The other poses, and all the measurements connected to them, are not marginalised out as in the filter, but simply discarded — they do not contribute to estimates. Compared to filtering, this approach will produce a graph which has more elements (since many past poses are retained), but importantly for inference the lack of marginalisation means that it will remain sparsely inter-connected. The result is that graph optimisation remains relatively efficient, *even if the number of features in the graph and measured from the keyframes is very high.* The ability to incorporate more feature measurements counters the information lost from the discarded frames. So the key question is whether it makes sense to summarise the information gained from historic poses and measurements by joint probability distributions in state space and propagate these through time (filtering), or to discard some of those measurements in such a way that repeated optimisation from scratch becomes feasible (keyframe BA), and propagating a probability distribution through time is unnecessary.

4.2 Experimental Design

Hence, there are two main classes of real-time visual SLAM systems capable of consistent local mapping. The first class is based on filtering. Early approaches based on the EKF were developed by Chiuso et al. (2002), Jung & Lacroix (2003) and Davison (2003). Several enhancements — mainly improving the parametrisation — were suggested afterwards (e.g. Montiel et al., 2006; Pietzsch, 2008; Civera et al., 2009b). Probably the best representative of this class is the approach of Eade &

Drummond (2007) which builds a map of locally filtered sub-maps. The other class is based on keyframe BA, mainly dominated by Klein & Murray's *Parallel Tracking and Mapping* (PTAM, 2007) framework (but also highly related to the BA-based visual odometry approaches discussed in the previous chapter).

For defining an experimental setup, we keep the two successful representatives, PTAM and Eade & Drummond's system, in mind. These systems are similar in many regards, incorporating parallel processes to solve local metric mapping, appearance-based loop closure detection and background global map optimisation over a graph. They are very different at the very local level, however, in exactly the way that we wish to investigate, in what constitutes the fundamental building block of their mapping processes. In PTAM, it is the keyframe, a historical pose of the camera where a large number of features are matched and measured. Only information from these keyframes goes into the final map. All other frames are used locally for tracking but that information is ultimately discarded. Klein & Murray's key observation which permits real-time operation is that BA over keyframes does not have to happen at frame-rate. In their implementation, BA runs in one thread on a multi-core machine, completing as often as possible, while a second tracking thread does operate at frame-rate with the task of pose estimation of the current camera position with respect to the fixed map defined by the nearest keyframe. In Eade & Drummond's system, the building block is a 'node', which is a filtered probabilistic sub-map of the locations of features. Measurements from all frames are digested in this sub-map, but the number of features it contains is consequently much smaller. The spacing of keyframes in PTAM and Eade & Drummond's nodes is decided automatically in both cases, but turns out to be similar. Essentially, during a camera motion between two neighbouring keyframes or nodes, a high fraction of features in the image will remain observable. So in our simulations, we aim to isolate this very local part of the general mapping process: the construction of a building block which is a few nodes or the motion between a few keyframes. Thus, we wish to analyse both accuracy and computational cost. As a measure of accuracy, we consider only the error between the start and end point of a camera motion. This is appropriate as it measures how much camera uncertainty grows with the addition of each building block to a large map.

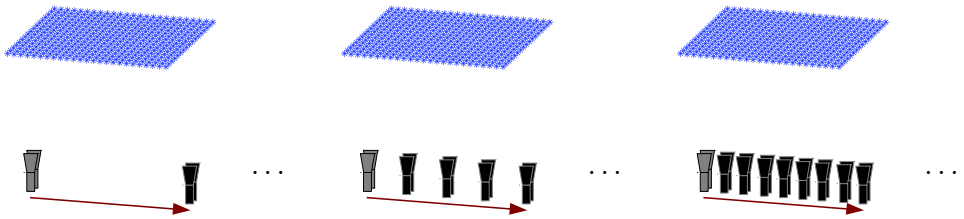
4.3 Preliminary Experiment

Before getting deep into a complex evaluation scheme by regarding the full SLAM pipeline, specific BA and filter variants, implementation subtleties, state space parametrisations as well as the connection between computational cost and accuracy, we will consider a simplified experimental setup.

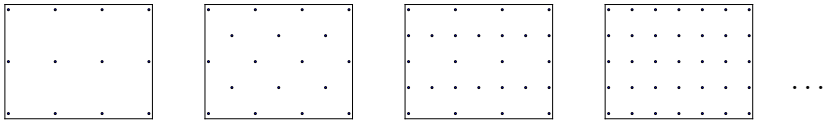
4.3.1 Setup and Problem Formulation

We simulate a scenario in which a stereo camera performs a short sideway motion (see Figure 4.2(a)). The length of the camera trajectory is one metre and a bounded fronto-parallel planar object at three metres depth is visible in the scene. The object is fully observable from all intermediate frames. This scenario is motivated twofold: Firstly, it represents a situation of relatively detailed local scene reconstruction, essentially optimising the local environment of one view with the support of very nearby surrounding views, as might be encountered practically for instance in small scale augmented reality, or object model reconstruction. Secondly, the estimation produced in this setting could be seen as a building block of a sub-mapping SLAM system. In particular it is very comparable to a single filter node of Eade & Drummond's SLAM framework.

We choose a camera with a resolution of 640×480 pixels and a focal length of $f = 500$. Thus, the simulated camera has a horizontal view angle of $65.2°$ and a vertical view angle of $51.3°$. The baseline between the stereo camera pair is set to 10cm. Let us assume that our camera captures a number of 36 frames during the one meter trajectory — in addition to the initial frame which is captured at the reference pose $\mathtt{T}_0$. Furthermore, we assume that the planar object consists of a number of 425 regularly arranged feature points. Thus, there are two essential parameters to vary: First, we select the number of keyframes M out of the range $\{1, 2, \dots 36\}$. In the case of $M = 1$, only the first frame at $\mathtt{T}_0$ and the last frame at $\mathtt{T}_M$ are used for the SLAM estimate. Second, we vary the number the number of features N in the range $\{12, \dots 425\}$. In particular, we select the number of points out of the regular pattern in such a way that a pattern with a higher density always includes all points from a pattern with lower density — as illustrated Figure 4.2(b). Our objective is to maximize the accuracy of the estimated motion. In particular,



a) sideways motion with varying number of frames M



b) planar scenes with varying number of points N

Figure 4.2: Preliminary experiment. (a) shows a stereo camera moving sideways in front of a planar scene. (b) shows the planar scene which consists of 12 to 425 points arranged in a regular pattern.

we measure the accuracy of the final camera pose $\mathtt{T}_M$ with respect to the initial reference pose $\mathtt{T}_0$.

In our SLAM problem, we seek to estimate the state vector $\mathbf{x}$ over the joint state of all camera poses $\mathtt{T}_i$ and scene points $\mathbf{y}_j$. In a least squares formulation, we wish minimize the cost χ^2,

$$\chi^2(\mathbf{x}) = \sum_{i=0}^{M}\sum_{j=1}^{N}(\mathbf{z}_{i,j} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}_j))\Sigma_{\mathbf{z}_{i,j}}^{-1}(\mathbf{z}_{i,j} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}_j)) \ , \tag{4.1}$$

with respect to $\mathbf{x} = (\mathtt{T}_1, \ldots, \mathtt{T}_M, \mathbf{y}_1, \ldots, \mathbf{y}_n)$. Here, $\hat{\mathbf{z}}$ is the stereo forward model as specified in Section 3.2.2. The first pose $\mathtt{T}_0$ is fixed and defines the reference frame. The measurement noise is set to a standard deviation of $\sigma = \frac{1}{2}$ pixels:

$$\Sigma_{\mathbf{z}} = \begin{bmatrix} \Sigma_{1,1} & & \\ & \ddots & \\ & & \Sigma_{M,N} \end{bmatrix} = \begin{bmatrix} \sigma^2 & & \\ & \ddots & \\ & & \sigma^2 \end{bmatrix} . \tag{4.2}$$

4.3.2 Accuracy Analysis using Entropy Reduction

The accuracy of our SLAM problem for a specific setting – M poses and N points – can be estimated without the need of minimizing χ^2 explicitly. Starting form the ground truth, we can estimate the uncertainty $\Sigma_{\mathbf{x}}$ over the joint SLAM state given the measurement uncertainty $\Sigma_{\mathbf{z}}$ using covariance back-propagation:

$$\Sigma_{\mathbf{x}} = (\mathtt{J}^\top \Sigma_{\mathbf{z}}^{-1} \mathtt{J})^{-1} \ , \tag{4.3}$$

with $\mathtt{J}$ being the Jacobian of the least-squares cost (4.1). We are only interested in the 6×6 sub-matrix $\Sigma_{\mathtt{T}}$ which specifies the uncertainty of the final camera pose $\mathtt{T}_M$,

$$\Sigma_{\mathtt{T}} = \begin{bmatrix} \Sigma_{\boldsymbol{\upsilon}} & \Sigma_{\boldsymbol{\omega},\boldsymbol{\upsilon}}^\top \\ \Sigma_{\boldsymbol{\omega},\boldsymbol{\upsilon}} & \Sigma_{\boldsymbol{\omega}} \end{bmatrix} \ . \tag{4.4}$$

We are analysing the translational uncertainty $\Sigma_{\boldsymbol{\upsilon}}$ and the rotational uncertainty $\Sigma_{\boldsymbol{\omega}}$ independently. In this way, we avoid the ill-posed question of forming a single unified measure representing both rotation and translation accuracy.

Finally, we analyse the influence of different parameter combinations $\langle M, N \rangle$ in terms of *entropy reduction*. The differential entropy of a multivariate Gaussian $X = \langle \boldsymbol{\mu}_X, \Sigma_X \rangle$ is defined as:

$$H(X) = \frac{1}{2} \log_2((2\pi e)^N \det(\Sigma_X)) \ . \tag{4.5}$$

Now, the relative difference between two Gaussians $X = \langle \boldsymbol{\mu}_X, \Sigma_X \rangle$, $Y = \langle \boldsymbol{\mu}_Y, \Sigma_Y \rangle$ can be described using the difference of entropy:

$$E(X, Y) := H(X) - H(Y) \ . \tag{4.6}$$

This measure is only meaningful if both distributions share (at least approximatively) the same mean. If $H(X) > H(Y)$ it can be seen as a entropy reduction measure: How much more accuracy do we gain, if we do Y instead of X. It holds that

$$\begin{aligned} E(X, Y) &= H(X) - H(Y) && (4.7) \\ &= \frac{1}{2} \log_2((2\pi e)^N \det(\Sigma_X)) - \frac{1}{2} \log_2((2\pi e)^N \det(\Sigma_Y)) && (4.8) \\ &= \frac{1}{2} \log_2 \left(\frac{\det(\Sigma_X)}{\det(\Sigma_Y)} \right) . && (4.9) \end{aligned}$$

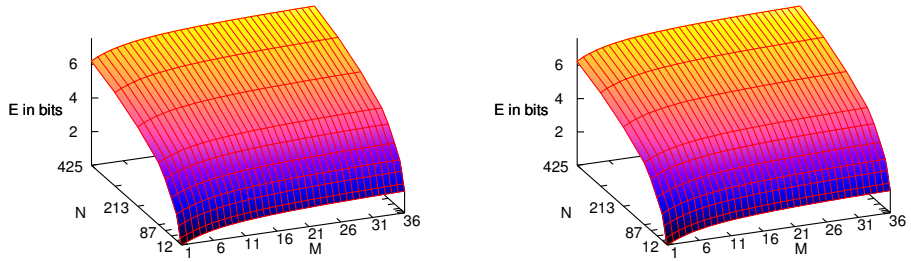


Figure 4.3: Result of the preliminary experiment. The surfaces show the rotational and translational accuracy for varying number of keyframes M and points N in terms of entropy reduction in bit (the higher, the better).

Here, we use this relative entropy measure in order to compare the general setting $\langle M, N\rangle$ to the minimal setting $\langle 1, 12\rangle$ where only one end pose and 12 points are used for the SLAM estimate. Note that both means $\boldsymbol{\mu}_{\langle 1,12\rangle} = \boldsymbol{\mu}_{\langle N,M\rangle}$ are set to be the ground truth. Thus, we compute how much accuracy we gain, i.e. how much entropy is reduced

$$E = \frac{1}{2}\log_2\left(\frac{\det(\Sigma_{\langle 1,12\rangle})}{\det(\Sigma_{\langle M,N\rangle})}\right), \tag{4.10}$$

if we use M keyframes and N points instead of the minimal setting, with $\Sigma_{\langle i,j\rangle}$ being both either 3×3 covariance matrices of the final camera translation Σ_υ or covariance matrices of the final camera rotation Σ_ω. Geometrically, the measure E describes the ratio of the volumes of the two ellipsoids $\Sigma_{\langle 1,12\rangle}$ and $\Sigma_{\langle M,N\rangle}$ on a log scale. For numerical stability, the natural logarithms of the absolute values of the determinants are calculated directly, subtracted and normalised afterwards:

$$E = \frac{1}{2\ln(2)}(\ln|\det(\Sigma_{\langle 1,12\rangle})| - \ln|\det(\Sigma_{\langle M,N\rangle})|) \tag{4.11}$$

Here, we exploit the fact that the determinant of a covariance matrix is always positive.

4.3.3 Preliminary Results

The influence of the parameters $\langle M, N\rangle$ is illustrated in Figure 4.3. As can be seen, increasing the number of features significantly increases the accuracy. On the other

hand, increasing the number of intermediate frames has only a minor influence. At each point on the accuracy surface it is more beneficial to double the number of points N instead of doubling the number of intermediate frames M in order to maximise the accuracy. This is true for accuracy measured on the rotational as well as on the translational component of the pose. Comparing the cost of BA (linear in N) to the cost of filtering (cubic in N), it becomes clear that BA is the more efficient technique — especially if high accuracy is required. This is the most important result of our analysis.

So far, this result relies on a number of assumptions: First, we merely analysed the accuracy of SLAM using uncertainty propagation. We did not analyse the absolute accuracy and cost of full incremental SLAM pipeline (including pose tracking and feature initialisation). Especially, we have not taken the well-known effect into account that the accuracy of filtering can degrade from the maximum likelihood estimate due to linearisation issues (Julier & Uhlmann, 2001; Castellanos et al., 2004). Second, we only analysed stereo SLAM and did not investigate the specifics of monocular vision. Finally, we assumed that all points are visible in all frames, which is an idealisation of a typical camera path where there is only a partial scene overlap between the first and the last frame. In the following we will lift these assumptions and see that the results of the preliminary experiment can be confirmed under more general conditions.

4.4 Bundle Adjustment and Filter Variants

In our main comparison, we want to analyse the difference between BA and filtering in a series of Monte Carlo experiment. Therefore, we need to decide on specific implementations.

4.4.1 State Prediction and Motion Prior

EKF-based SLAM consists of two phases: The *state prediction* and the *update step*. Let $\langle \boldsymbol{\mu}_{t-1}, \Sigma_{t-1} \rangle$ denote the mean and covariance of the EKF state space. Furthermore, let $g(\cdot, \mathbf{u}_{t-1})$ be a state prediction model, and $\mathbf{u}_{t-1}$ some control input. For instance, $\mathbf{u}_{t-1}$ could be the output of a wheel odometer and g the corresponding

motion model. Then the new state $\langle \bar{\boldsymbol{\mu}}_t, \bar{\Sigma}_t \rangle$ can be predicted as:

$$\bar{\boldsymbol{\mu}}_t = g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_{t-1}) \,, \tag{4.12}$$

$$\bar{\Sigma}_t = \frac{\partial g}{\partial \boldsymbol{\mu}_{t-1}} \Sigma_{t-1} \frac{\partial g}{\partial \boldsymbol{\mu}_{t-1}}^\top + \frac{\partial g}{\partial \mathbf{u}_{t-1}} \Sigma_{\mathbf{u}_{t-1}} \frac{\partial g}{\partial \mathbf{u}_{t-1}}^\top \text{ (Thrun et al., 2005, p.59)} \tag{4.13}$$

In visual SLAM, no external odometry measurements are available. In MonoSLAM, Davison et al. (2003; 2007) incorporate a constant velocity motion model. Thus, they calculate a *prior distribution* $\langle \bar{\boldsymbol{\mu}}_t, \bar{\Sigma}_t \rangle$ before any visual measurement is integrated in the update step. This works well as long as the smooth motion assumption is not violated. However, if there are rapid changes in the motion, such a prior could potentially introduce a strong bias in the estimation process. Probably, this is the reason why PTAM as well as the filtering approach of Eade & Drummond do not incorporate a motion prior, but assume a uniform prior instead (see also Section 2.3.4). Therefore, we will use BA and filter variants with uniform priors on the camera poses.

4.4.2 Implementation for Bundle Adjustment and Filtering

For BA, we apply a state of the art approach using the Schur-Complement, and a sparse Cholesky solver. It is less obvious what kind of filter variant to use. The standard EKF is fundamentally different from the BA formulation of SLAM, but has well-known limitations. However, there is a broad middle ground between filtering and BA/smoothing (which we will discuss in Section 4.7.1). Indeed, if one tries to define the best possible filter by modifying the standard approach, one would converge more and more towards BA. Therefore, it is important to define precisely what we understand by a filter. Our concept of a filter is a cluster of related properties:

1. *Explicit representation of uncertainty*: A set of parameters is represented using a multivariate normal distribution.

2. *Marginalisation*: Temporary/outdated parameters are marginalised out in order to keep the state representation compact.

3. *Cheap access to covariance*: The joint covariance can be recovered from the filter representation without increasing the overall algorithmic complexity.

It is obvious that property 1 is the core property of the (Gaussian) filter concept. Property 2 is very common in visual SLAM, since filters are often applied at frame-rate. Each single frame produces a new pose estimate. In order to avoid an explosion in the state space, past poses are marginalised out. Still, property 3 is a crucial characteristic which distinguishes BA from filtering: It is possible to calculate the covariance of the BA problem using covariance propagation, but this would increase the algorithmic complexity of BA significantly.

There are two fundamentally different approaches of Gaussian filters. The standard approach is the EKF, which represents uncertainty using a covariance matrix Σ. It is easy to see that the EKF fulfils all the three properties defined above. Its dual is the extended information filter which represent the uncertainty using the inverse covariance or information matrix $\Lambda = \Sigma^{-1}$. In the SLAM community, the EKF and its variants are particular popular since its computational complexity is $O(K^2)$ while it is in general $O(K^3)$ for the information filter, with K being the total number of features in the map. Since we only consider a local building block of SLAM the computational complexity is dominated by the number of visible features N, leading to a complexity of $O(N^3)$ for both filter types. Thus, both approaches are largely equivalent for our purposes. Indeed we choose the information matrix representation. The reasons are twofold: First, the information filter approach is conceptually more appropriate for our comparison since the relation between filtering and BA becomes more obvious — both are non-linear least-squares methods. Second, the information form allows us to include variables without any prior on the state space. Thus, we can include new poses without any motion prior, and also we are able to represent infinite depth uncertainty for monocular inverse depth features. In particular, we follow Eade & Drummond (2007) as well as Sibley et al. (2005) and employ the *Gauss-Newton filter* (see Section 2.3.4). It iteratively solves the normal equations using the Cholesky method and therefore is the dual of the iterative EKF (Bell & Cathey, 1993).

Furthermore, note that the Gauss-Newton filter is equivalent to the correction step of the classic Square Root Information Filter (SRIF, Dyer & McReynolds, 1969). The SRIF never constructs the normal equations explicitly and solves the problem using an orthogonal decomposition on the square root form. While performing Gauss-Newton using Cholesky decomposition is less numerically stable than performing the orthogonal decomposition method, it is computationally more effi-

Algorithm 3 BA-SLAM pipeline

$\mathcal{X}$:= initialise_points($\mathcal{Z}_0$)
for each keyframe/time step $i = 1$ to M **do**
  if a number of $n \geq 1$ points left field of view **then**
    $\mathcal{Y}$:= $\mathcal{Y}$ $\cup$ initialise_n_new_points($\mathcal{Z}_i, n$)
  end
  $\mathtt{T}_i$:= motion_only_BA}($\mathcal{Y}, \mathcal{Z}_i$)
  $\mathcal{Y}$:= structure_only_BA($\mathtt{T}_{0:i}, \mathcal{Y}, \mathcal{Z}_{0:i}$)
  $\mathtt{T}_{1:i}, \mathcal{Y}$:= full_BA($\mathtt{T}_{1:i}, \mathcal{Y}, \mathcal{Z}_{0:i}$)
end

cient and therefore the standard approach for real-time least-squares problems nowadays. Indeed, a sufficient numerical stability of even rank-deficient problems can be archived by applying a robust variant of Cholesky — such as the pivoted $\mathtt{L}^\top\mathtt{DL}$ decomposition used in the *Eigen* matrix library[2] — and the Levenberg-Marquardt damping term, called *Tikhonov regularisation* (Tikhonov & Arsenin, 1977) in this context.

4.5 Implementation of Visual SLAM

In this section, we describe the concrete implementation we used for BA and filter-SLAM. We first focus on stereo SLAM. The specifics of monocular SLAM are discussed afterwards.

4.5.1 BA-SLAM

In BA, we optimise simultaneously for structure and motion by minimising the reprojection error:

$$\chi^2(\mathbf{x}) = \sum_{\mathbf{z}_{i,j} \in \mathcal{Z}_{0:i}} (\mathbf{z}_{i,j} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}_j))^2 \tag{4.14}$$

with respect to $\mathbf{x} = (\mathtt{T}_1, ..., \mathtt{T}_i, \mathcal{Y})^\top$ with $\mathcal{Y}$ being the set of all points $\mathbf{y}_j$. The first frame $\mathtt{T}_0$ is typically fixed in order to eliminate the underlying gauge freedom. We employ the standard approach to BA as discussed in Section 3.3.4, and we implement it using the **g2o** framework (Kümmerle et al., 2011a).

[2] http://eigen.tuxfamily.org/dox/TutorialLinearAlgebra

BA is just the core of the full SLAM pipeline. We use the following scheme which is summarised in Table 3. In the first frame, we initialise the 3D points $\mathbf{y}_j \in \mathcal{Y}$ from the set of initial measurements $\mathcal{Z}_0$. As described in Section 3.4.3, we select a set $\mathcal{X}$ of N points from a larger set of scene point candidates using a quadtree to ensure that the corresponding 2D observations $\mathbf{z}_{0,j} \in \mathcal{Z}_0$ are spread approximately equally across the image. For each time step, that is for each new keyframe, four steps are performed. First, we optionally initialise new 3D points in case some old features have left the field of view. Using the quadtree, we initialise new points where the feature density is low. Second, the current pose $\mathtt{T}_i$ is estimated using *motion-only BA*. Thus, we minimise the reprojection error:

$$\chi^2(\mathtt{T}_i) = \sum_{\mathbf{z}_j \in \mathcal{Z}_i} (\mathbf{z}_j - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}_j))^2 \tag{4.15}$$

with respect to the current camera $\mathtt{T}_i$. We simply initialise the current pose to the previous pose $\mathtt{T}_i = \mathtt{T}_{i-1}$, though one could also use a motion model to predict a better initial guess.[3] Third, we perform *structure-only BA* by minimising

$$\chi^2(\mathcal{Y}) = \sum_{\mathbf{z}_{i,j} \in \mathcal{Z}_{0:i}} (\mathbf{z}_{i,j} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}_j))^2 \tag{4.16}$$

with respect to the set of points $\mathcal{Y}$. Finally, we perform joint optimisation of structure and motion as formalised in equation (4.14).

4.5.2 Filter-SLAM

For filtering, it is especially important that the state representation is as 'linear' as possible. It proved to be useful, especially but not exclusively for monocular SLAM, to represent 3D points using anchored inverse depth coordinates (Civera et al., 2008). Our effort is to combine the most successful approaches. We represent points using the inverse depth formulation of Eade (2008). As in Pietzsch (2008), the bundle of points, which were initialised at the same time, are associated with its common anchor pose $\mathtt{T}_{a(j)}$; $a(j) = k$ is a function which assigns an anchor frame index k for each point index j. We use the inverse depth representation as described

[3] Note that one can integrate a motion model without enforcing a motion prior. For instance, PTAM uses a constant-velocity motion model to calculate the initial pose for motion-only optimisation. The optimisation, however, is free to depart from this initial guess in order to minimise the reprojection error without any penalty cost.

in Section 3.4.4. Thus, the reprojection error of a point $\boldsymbol{\psi}_j$ in frame $\mathbf{T}_i$ is

$$\mathbf{d}_{i,j} := \mathbf{z}_i - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathtt{T}_{a(j)}^{-1} \Pi(\boldsymbol{\psi}_i)) \quad \text{with} \quad \Pi(\mathbf{a}) = \frac{1}{a_3} \begin{pmatrix} a_1 \\ a_2 \\ 1 \end{pmatrix} \tag{4.17}$$

As motivated above, we perform filtering using a Gauss-Newton filter. Thus, we minimize the following sum of squares function,

$$\chi^2(\boldsymbol{\Phi}_i, \mathtt{T}_i) = (\boldsymbol{\Phi}_i \ominus \boldsymbol{\Phi}_{i-1})^\top \Lambda_{\boldsymbol{\Phi}_{i-1}} (\boldsymbol{\Phi}_i \ominus \boldsymbol{\Phi}_{i-1}) + \sum_{\mathbf{z}_j \in \mathcal{Z}_i} \mathbf{d}_{i,j}^\top \Lambda_{\mathbf{z}} \mathbf{d}_{i,j} \ , \tag{4.18}$$

with respect to the map $\boldsymbol{\Phi}_i$ and the current camera pose $\mathtt{T}_i$. Here, $\langle \boldsymbol{\Phi}_{i-1}, \Lambda_{\boldsymbol{\Phi}_{i-1}} \rangle$ is the Gaussian map prior. Differences between two poses are calculated in the tangent space of $\mathbf{SE}(3)$:

$$\mathtt{T}^{[i]} \ominus \mathtt{T}^{[i-1]} := \log \left(\mathtt{T}^{[i]} \cdot \left(\mathtt{T}^{[i-1]} \right)^{-1} \right)_{\mathfrak{se}(3)}^{\vee} \ , \tag{4.19}$$

while the difference between inverse depth points is simply standard subtraction $\boldsymbol{\psi}^{[i]} \ominus \boldsymbol{\psi}^{[i-1]} := \boldsymbol{\psi}^{[i]} - \boldsymbol{\psi}^{[i-1]}$. Since we do not impose a motion prior on $\mathtt{T}_i$, the prior joint information over the current pose $\mathtt{T}_i$ and the map $\boldsymbol{\Phi}_{i-1}$ is

$$\Lambda_{i-1} := \begin{pmatrix} \Lambda_{\boldsymbol{\Phi}_{i-1}} & \Lambda_{\boldsymbol{\Phi}_{i-1},\mathtt{T}_i}^\top \\ \Lambda_{\boldsymbol{\Phi}_{i-1},\mathtt{T}_i} & \Lambda_{\mathtt{T}_i} \end{pmatrix} = \begin{pmatrix} \Lambda_{\boldsymbol{\Phi}_{i-1}} & \mathtt{O}_{3n \times 6} \\ \mathtt{O}_{6 \times 3n} & \mathtt{O}_{6 \times 6} \end{pmatrix} . \tag{4.20}$$

Following Section 2.3.3, we calculate the update of the information matrix:

$$\Lambda_i = \Lambda_{i-1} + \mathtt{D}^\top \begin{bmatrix} \Sigma_{\mathbf{z}}^{-1} & & \\ & \ddots & \\ & & \Sigma_{\mathbf{z}}^{-1} \end{bmatrix} \mathtt{D} \ . \tag{4.21}$$

$\mathtt{D}$ is the sparse Jacobian of the stacked reprojection function:

$$\mathbf{d} = (\mathbf{d}_1^\top, ..., \mathbf{d}_N^\top)^\top \tag{4.22}$$

with respect to the pose $\mathtt{T}_i$, to the points $\boldsymbol{\psi}_1, ..., \boldsymbol{\psi}_N$ and to the corresponding anchor frames $\{\mathtt{T}_{a(j)} | j = 1, ..., N\}$. Details of this uncertainty propagation are in Algorithm 4.

The whole filter-SLAM pipeline is sketched in Table 5. In the first frame, the inverse depth points $\boldsymbol{\psi}$ are initialised from the stereo observation $\mathbf{z}_s$:

$$\boldsymbol{\psi} = \begin{pmatrix} \frac{u_l - p_u}{f} & \frac{v_l - p_v}{f} & \frac{u_l - u_r}{fb} \end{pmatrix}^\top \ , \tag{4.23}$$

Algorithm 4 Uncertainty propagation

for each point $\boldsymbol{\psi}_j$ in $\mathcal{X}$ **do**
  $\mathtt{J}_{\mathtt{T}} := \left.\frac{\partial \mathbf{d}(\exp(\boldsymbol{\epsilon})\cdot\mathtt{T},\mathtt{T}_{a(j)},\boldsymbol{\psi}_j)}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0}$ //*pose Jacobian*
  $\mathtt{J}_{\mathtt{T}} := \frac{\partial \mathbf{d}(\mathtt{T},\mathtt{T}_{a(j)},\boldsymbol{\psi}_j)}{\partial \boldsymbol{\psi}_j}$ //*point Jacobian*
  $\Lambda_{\mathtt{T}_i,\mathtt{T}_i} := \Lambda_{\mathtt{T}_i,\mathtt{T}_i} + \mathtt{J}_{\mathtt{T}}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\mathtt{T}}$ //*update pose block*
  $\Lambda_{\mathtt{T}_i,\boldsymbol{\psi}_j} := \Lambda_{\mathtt{T}_i,\boldsymbol{\psi}_j} + \mathtt{J}_{\mathtt{T}}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\psi}$ //*update pose-point blocks*
  $\Lambda_{\boldsymbol{\psi}_j,\mathtt{T}_i} := \Lambda_{\boldsymbol{\psi}_j,\mathtt{T}_i} + \mathtt{J}_{\boldsymbol{\psi}}^\top \Sigma_{\mathbf{z}} \mathtt{T}_{\mathtt{T}}$
  $\Lambda_{\boldsymbol{\psi}_j,\boldsymbol{\psi}_j} := \Lambda_{\boldsymbol{\psi}_j,\boldsymbol{\psi}_j} + \mathtt{J}_{\boldsymbol{\psi}}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\boldsymbol{\psi}}$ //*update point-point blocks*
  if $a(j) > 1$ **then**
    $\mathtt{J}_{\mathtt{T}_a} := \left.\frac{\partial \mathbf{d}(\mathtt{T},\exp(\boldsymbol{\epsilon})\mathtt{T}_{a(j)},\boldsymbol{\psi}_j)}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=0}$ //*anchor pose Jacobian*
    $\Lambda_{\mathtt{T}_{a(j)},\mathtt{T}_{a(j)}} := \Lambda_{\mathtt{T}_{a(j)},\mathtt{T}_{a(j)}} + \mathtt{J}_{\mathtt{T}_a}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\mathtt{T}_a}$ //*update anchor block*
    $\Lambda_{\mathtt{T}_{a(j)},\boldsymbol{\psi}_j} := \Lambda_{\mathtt{T}_{a(j)},\boldsymbol{\psi}_j} + \mathtt{J}_{\mathtt{T}_a}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\boldsymbol{\psi}}$ //*update anchor-point blocks*
    $\Lambda_{\boldsymbol{\psi}_j,\mathtt{T}_{a(j)}} := \Lambda_{\boldsymbol{\psi}_j,\mathtt{T}_{a(j)}} + \mathtt{J}_{\boldsymbol{\psi}}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\mathtt{T}_a}$
    $\Lambda_{\mathtt{T}_i,\mathtt{T}_{a(j)}} := \Lambda_{\mathtt{T}_i,\mathtt{T}_{a(j)}} + \mathtt{J}_{\mathtt{T}}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\mathtt{T}_a}$ //*update pose-anchor blocks*
    $\Lambda_{\mathtt{T}_{a(j)},\mathtt{T}_i} := \Lambda_{\mathtt{T}_{a(j)},\mathtt{T}_i} + \mathtt{J}_{\mathtt{T}_a}^\top \Sigma_{\mathbf{z}} \mathtt{J}_{\mathtt{T}}$
  end
end

with f being the focal length and b being the baseline of the stereo camera. As a side note, this formula highlights the close relationship between inverse depth $\psi_3 = \frac{u_l - u_r}{fb}$ and stereo disparity $u_l - u_r$. We initialise the corresponding information matrix as

$$\Lambda_\psi = \left(\frac{\partial \mathbf{z}_s}{\partial \psi} \Sigma_{\mathbf{z}} \frac{\partial \mathbf{z}_s}{\partial \psi}^\top \right)^{-1} \quad \text{with} \quad \frac{\partial \mathbf{z}_s}{\partial \psi} = \begin{pmatrix} \frac{1}{f} & 0 & 0 \\ 0 & \frac{1}{f} & 0 \\ \frac{1}{fb} & 0 & -\frac{1}{fb} \end{pmatrix} . \tag{4.24}$$

At each time step i, we do the following: First, we decide whether we want initialise new points. If this is the case, we define the previous estimated pose $\mathtt{T}_{i-1}$ as the new anchor frame $\mathtt{T}_a$ and augment the map state accordingly $\boldsymbol{\Phi}_i = (\boldsymbol{\Phi}_{i-1}, \mathtt{T}_a)^\top$. Then, we marginalise out n old points from the filter state and replace them with n new points anchored to $\mathtt{T}_a$. As in BA-SLAM, a quadtree is used for point initialisation. Otherwise, we marginalise out the pose $\mathtt{T}_{i-1}$ from Λ_{i-1}

$$\Lambda_{\boldsymbol{\Phi}_{i-1}} = \Lambda_{\boldsymbol{\Phi}_{i-1}} - \Lambda_{\mathtt{T}_{i-1},\boldsymbol{\Phi}_{i-1}}^\top \Lambda_{\mathtt{T}_{i-1}}^{-1} \Lambda_{\mathtt{T}_{i-1},\boldsymbol{\Phi}_{i-1}} . \tag{4.25}$$

Next, we approximate the new camera pose $\mathtt{T}_i$ given the previous map $\boldsymbol{\Phi}_{i-1}$. In traditional filter-based SLAM implementations, this step is often omitted. However, in the case of large camera displacements (e.g. due to low filter frequency) it is

Algorithm 5 Filter-SLAM pipeline.

$\mathtt{T}_a$:= $\mathtt{T}_0$
$\langle \boldsymbol{\Phi}_0, \Lambda_{\boldsymbol{\Phi}_0} \rangle$:= Initialise map using equations (4.23) and (4.24).
for each time step $i = 1$ to M **do**
  if a number of $n \geq 1$ points left field of view **then**
    $\mathtt{T}_a$:= $\mathtt{T}_{i-1}$ *//old pose is a new anchor pose*
    $\boldsymbol{\Phi}_i$:= $\begin{pmatrix} \boldsymbol{\Phi}_{i-1} \\ \mathtt{T}_a \end{pmatrix}$ *//augment map with anchor pose*
    $\Lambda_{\boldsymbol{\Phi}_i}$:= Λ_{i-1}
    $\langle \boldsymbol{\Phi}_i, \Lambda_{\boldsymbol{\Phi}_i} \rangle$:= Marginalise out the n invisible points and initialise n new points anchored to $\mathtt{T}_a$.
  else
    $\boldsymbol{\Phi}_i$:= $\boldsymbol{\Phi}_{i-1}$
    $\Lambda_{\boldsymbol{\Phi}_i}$:= $\Lambda_{\boldsymbol{\Phi}_{i-1}} - \Lambda^{\top}_{\mathtt{T}_{i-1},\boldsymbol{\Phi}_{i-1}} \Lambda^{-1}_{\mathtt{T}_{i-1}} \Lambda_{\mathtt{T}_{i-1},\boldsymbol{\Phi}_{i-1}}$ *//marginalise out old pose*
  end
  $\Sigma_{\boldsymbol{\Phi}_i}$:= $\Lambda^{-1}_{\boldsymbol{\Phi}_i}$ *//calculate covariance, optionally*
  $\mathtt{T}_i$:= Calculate motion either using motion-only BA (4.15) or using a map prior $\langle \boldsymbol{\Phi}_i, \Sigma_{\boldsymbol{\Phi}_i} \rangle$ (Algorithm 6).
  $\begin{pmatrix} \boldsymbol{\Phi}_i \\ \mathtt{T}_i \end{pmatrix}$:= Joint filter update by minimising energy (4.18).
  Λ_i := Augment information matrix and update it (Algorithm 4).
end

desirable to approximate the camera motion before applying the joint filter update. There are two possibilities to estimate the camera pose given a known map. One can either do motion-only BA by minimising the cost (4.15). Here we assume that the points are accurately known. In the case that there is a significant uncertainty in the map, and a model of this uncertainty is available, we can do better. As shown in Algorithm 6 and described by Eade (2008, pp.126), we can estimate a pose given a Gaussian map prior. The effect is that taking account of the 3D uncertainty in point positions will weight their impact on camera motion estimation, and better accuracy will be obtained because accurately located points will be trusted more than uncertain ones. The pros and cons of these two approaches are analysed in Section 4.6.3. Finally, we perform the joint filter estimate and update the information matrix as discussed above.

Algorithm 6 Pose estimation given Gaussian map prior $\langle \Phi, \Sigma_\Phi \rangle$.

1 $\mathtt{S} := \mathtt{J}_\Phi(\mathtt{T})\Sigma_\Phi \mathtt{J}_\Phi(\mathtt{T})^\top + \Sigma_\mathbf{z}$ *//calculate innovation covariance*
2 $\chi^2 := \mathbf{d}(\Phi, \mathtt{T})^\top \mathtt{S}^{-1} \mathbf{d}(\Phi, \mathtt{T})$ *//calculate residual error*
3 **for** some iterations **do**
4 **repeat**
5 $\boldsymbol{\delta} := (\mathtt{J}_\mathtt{T}(\mathtt{T})^\top \mathtt{S}^{-1} \mathtt{J}_\mathtt{T}(\mathtt{T}) + \mu \mathtt{I})^{-1} \cdot (-\mathtt{J}_\mathtt{T}(\mathtt{T}) \mathtt{S}^{-1} \mathbf{d}(\Phi, \mathtt{T}))$
 //solve linear system
6 $\mathtt{T}_{\mathtt{new}} := \exp(\boldsymbol{\delta}) \cdot \mathtt{T}$ *//update pose*
7 $\mathtt{S} := \mathtt{J}_\Phi(\mathtt{T}_{\mathtt{new}}) \Sigma_\Phi \mathtt{J}_\Phi(\mathtt{T}_{\mathtt{new}})^\top + \Sigma_\mathbf{z}$ *//update innovation*
8 $\chi^2_{\mathtt{new}} := \mathbf{d}(\Phi, \mathtt{T}_{\mathtt{new}})^\top \mathtt{S}^{-1} \mathbf{d}(\Phi, \mathtt{T}_{\mathtt{new}})$ *//calculate new residual error*
9 **if** $\chi^2_{\mathtt{new}} \geq \chi^2$ **then**
10 Increase damping term μ.
11 **end**
12 **until** $\chi^2_{\mathtt{new}} < \chi^2$
13 $\mathtt{T} \leftarrow \mathtt{T}_{\mathtt{new}}$
14 $\chi \leftarrow \chi_{\mathtt{new}}$
15 Decrease damping term μ.
16 **end**

Here, $\mathtt{J}_\Phi(\mathtt{T}) := \frac{\partial \mathbf{d}(\Phi, \mathtt{T})}{\partial \Phi}$ and $\mathtt{J}_\mathtt{T}(\mathtt{T}) := \frac{\partial \mathbf{d}(\Phi, \exp(\boldsymbol{\epsilon}) \cdot \mathtt{T})}{\partial \boldsymbol{\epsilon}}|_{\epsilon = \mathtt{0}}$. In an approximative version, step 7 is skipped so that $\mathtt{S}^{-1}$ needs only to be calculated once.

4.5.3 Monocular SLAM

Monocular Bundle Adjustment

In Section 3.3.2 we learned that the gauge freedom of bundle adjustment increases from 6 DoF to 7 DoF if one moves from stereo to monocular vision. Even after fixing the origin $\mathtt{T}_0$, one dimension of scale gauge remains. We simply leave this one degree unfixed, since the damping term of Levenberg-Marquardt acts as a Tikhonov regulariser and can deal with gauge freedom effectively (Jeong et al., 2010). In BA-SLAM, new 3D points are triangulated between two consecutive keyframes using a set of independent filters as described in Section 3.4.4.

Monocular Filter

Since an anchored inverse depth representation was chosen for the filter, no substantial improvements are necessary when moving from stereo to monocular vision. As opposed to monocular BA, the monocular filter does not introduce a scale ambiguity.

The reason is that a non-trivial map distribution $\langle \boldsymbol{\Phi}, \boldsymbol{\Lambda}_{\boldsymbol{\Phi}} \rangle$ (with $\boldsymbol{\Lambda}_{\boldsymbol{\Phi}}$ having full rank) introduces a scale prior and therefore the degree of free gauge in equation (2.34) remains zero. This arbitrary scale factor is invented during bootstrapping (see below). For monocular vision, new features $\boldsymbol{\psi}$ are initialised with infinite uncertainty along the feature depth ψ_3:

$$\boldsymbol{\psi} = \begin{pmatrix} \frac{u-p_u}{f} & \frac{v-p_v}{f} & 1 \end{pmatrix}^\top \quad \text{and} \quad \boldsymbol{\Lambda}_{\boldsymbol{\psi}} = \operatorname{diag}\left(\frac{f^2}{\sigma_z^2}, \frac{f^2}{\sigma_z^2}, 0 \right) . \tag{4.26}$$

Structure and Motion Bootstrapping

Unless there is any additional prior knowledge such as a known object in the scene, monocular SLAM requires a special bootstrapping mechanism. We perform bootstrapping between three consecutive keyframes $\mathtt{T}_{b0}, \mathtt{T}_{b1}, \mathtt{T}_0$. The standard approach relies on the 5-point algorithm (Nistér, 2004), which however requires a RanSaC-like procedure. We instead employ an iterative optimisation, exploiting the fact that the consecutive keyframes share similar poses. First, we define $\mathtt{T}_{b0}$ as our fixed origin and apply monocular filtering between $\mathtt{T}_{b0}$ and $\mathtt{T}_{b1}$. Let us assume without loss of generality that $\mathtt{T}_{b0} = \mathtt{I}$. Note that now equation (2.34) has one dimension of gauge freedom, since there is infinite uncertainty along all feature depths ψ_3. This scale freedom during optimisation is handled with the LM damping term. Afterwards, we ensure the estimated motion $\mathtt{T}_{b1}$ has sufficient parallax. To summarise, we have estimated $6 + 3N$ parameters, while the underlying problem only has $5 + 3N$ DoF. In order to avoid a rank-deficient map distribution, we convert the pose $\mathtt{T}_{b1}$ into a 5 DoF representation by enforcing the additional constraint on $\mathbf{SE}(3)$ that the translation must be unity $|\mathbf{t}_{b1}| = 1$. First, we scale the whole state estimate — all inverse depth points $\boldsymbol{\psi}_j$ as well as the initial motion $\mathtt{T}_{b1}$ — such that $|\mathbf{t}_{b1}| = 1$. Afterwards, we perform uncertainty propagation (Algorithm 4) with a modified Jacobian $\mathtt{D}$ reflecting that the pose only has 5 DoF. Then, the 5 DoF pose is marginalised out. The resulting precision matrix $\boldsymbol{\Lambda}_{\boldsymbol{\Phi}}$ has full rank and enforces a scale prior (that the initial translation between $\mathtt{T}_{b0}$ and $\mathtt{T}_{b1}$ has unit length). Finally, we perform a standard monocular filter update (as described above) between frame $\mathtt{T}_{b1}$ and $\mathtt{T}_0$ so that the resulting map is well initialised and can be used for either BA-SLAM or filter-SLAM.

4.6 Experiments

Finally, we analyse the performance of BA-SLAM versus filter-SLAM by evaluating local motion in a set of Monte Carlo experiments.

4.6.1 Four Different Settings

In our final series of experiments, we consider four different scenes/motion patterns (see Figure 4.4). In Setting (i), the camera performs a motion sideways of 0.5 metre while observing an approximately planar scene. Here, all points are visible in all frames, and therefore the number of points in the map equals the number of observations per frame. The number M of keyframes (intermediate keyframes plus end frame, excluding the first frame) is varied between 1 and 16; more specifically $M \in \{1, 2, 4, 8, 16\}$. The number of observations N is chosen from $N \in \{15, 30, 60, 120, 240\}$. In addition, we also consider $N = 480$ for some specific cases.

The configuration of Setting (i) is designed following the preliminary experiment in Section 4.3.1. Since no new points need to be initialised, all points are anchored to the fixed origin $\mathtt{T}_0$. Setting (i) is very specific in the sense that all points are visible in all frames. In a typical visual odometry building block, there is only partial scene overlap. In each new frame of a sequence, some point projections leave the field of view while new points become visible. For Setting (ii), we have chosen a translation of 1.1m, so that the first and the final frame barely overlap. Therefore, at least one intermediate keyframe has to be used and we choose $M \in \{2, 4, 8, 16\}$. In Setting (iii), the camera performs a sideways motion plus rotation which leads to a partial scene overlap. Again, we choose $M \in \{2, 4, 8, 16\}$. In the final Setting (iv), the camera performs a sharp forward turn. This setting is typical for a camera mounted on a robot which performs a sharp 90° turn in an indoor environment. This setting is especially hard for Monocular SLAM: Scene points leave the field of view quickly while parallax is low due to the lack of translation. To achieve an acceptable level of robustness, we select $M \in \{4, 8, 16\}$.

For all optimisations (motion update, structure-only BA, full BA, joint filter update) we perform three LM iterations in Setting (i,ii) and ten LM iterations in Settings (iii,iv).

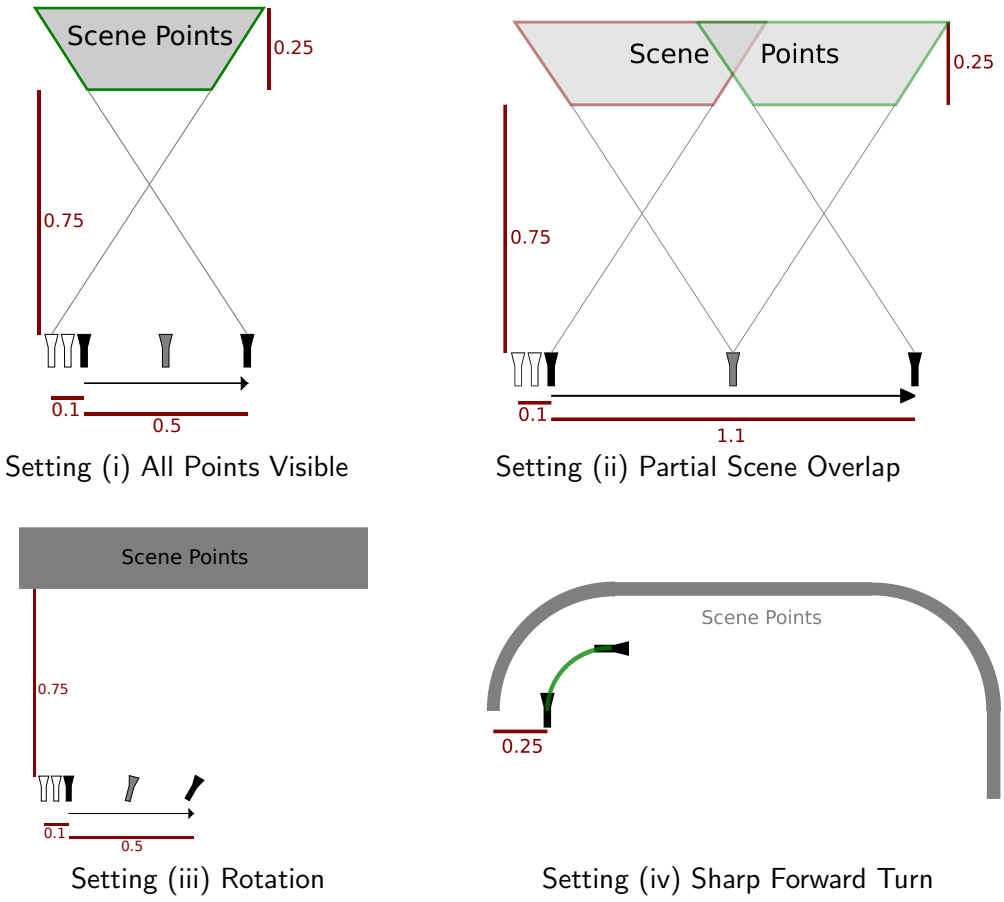
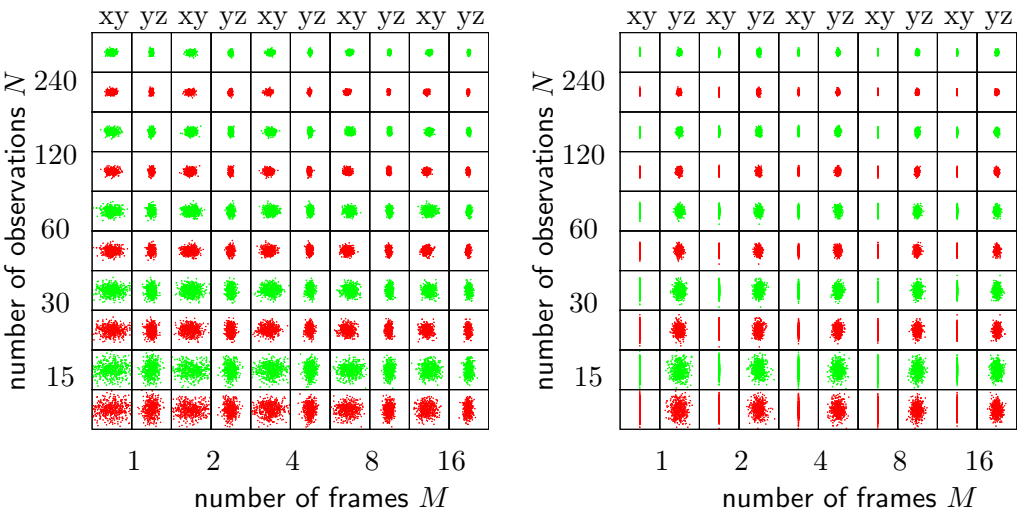


Figure 4.4: Birds-eye view of different motion/scene settings. Black cameras represent start and end pose. Intermediate poses are presented in gray. Unfilled cameras indicate the poses used for monocular bootstrapping $\mathtt{T}_{b0}, \mathtt{T}_{b1}$. Scene points are initialised within the gray-shaded areas. In Setting (i), all points are visible in all frames. In Setting (ii), there is only a partial scene overlap. Here, we illustrate the case with a single intermediate camera ($M = 2$). Some points are triangulated between the first and middle frames (right/red area), with others between middle and end frame (left/green area). In Setting (iii), the camera performs a 30° rotation while still moving sideways. In Setting (iv), the camera performs a sharp forward turn so that the scene points quickly leave the field of view. To avoid cluttering the figure, we do not show intermediate and bootstrapping poses here.



(a) Stereo SLAM (b) Monocular SLAM

Figure 4.5: End pose accuracy of stereo and monocular SLAM. Filtering results are shown in green (top rows), whereas BA results are shown in red (below). The distributions are shown in a zero-centred 1.5 cm sector.

4.6.2 Accuracy of Visual SLAM

We analyse the accuracy using the difference between the true final camera position $\mathbf{t}_{\text{true}}$ and the corresponding estimate $\mathbf{t}_{\text{est}}$:

$$\Delta\mathbf{t} = \mathbf{t}_{\text{true}} - \mathbf{t}_{\text{est}} \ . \tag{4.27}$$

For each chosen number of frames and points $\langle M, N \rangle$, we perform a set of $k = 500$ Monte Carlo trials. The sampling is performed over the measurement noise (with $\sigma = 0.5$ pixels) as well as the scene points (uniformly). For Setting (i) using stereo SLAM, the resulting plots are shown in Figure 4.5(a). Approximately, the presented discrete error distributions appear to consist of samples from unimodal, zero-mean Gaussian-like distributions.

In the case of monocular SLAM, we can only estimate the translation modulo an unknown scale factor. Therefore, we eliminate the scale ambiguity in our evaluation by normalising the estimated translation to the true scale:

$$\mathbf{t}^* = \frac{|\mathbf{t}_{\text{true}}|}{|\mathbf{t}_{\text{est}}|}\mathbf{t}_{\text{est}}. \tag{4.28}$$

Hence, all normalised estimates $\mathbf{t}^*$ lie on the sphere of radius $|\mathbf{t}_{\text{true}}|$. This explains why the projection of the error distribution onto the xy plane is elongated, with no

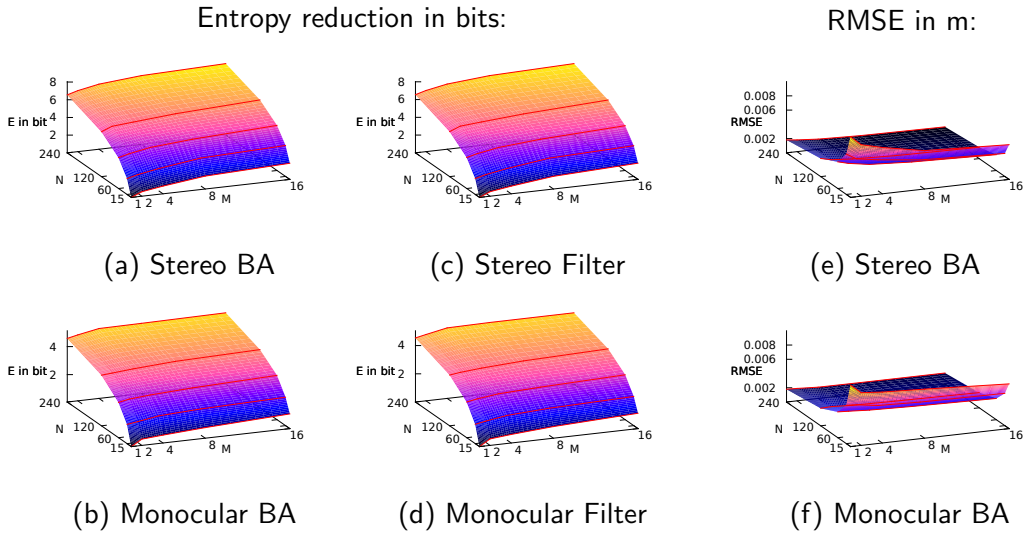


Figure 4.6: **Setting (i)**. Accuracy plots in terms of entropy reduction in bits and RMSE.

uncertainty along the unknown scale dimension (here x-axis). Interestingly, error distributions in the yz plane for monocular and stereo SLAM are of similar shape and size. In order to have a minimal and Gaussian-like parametrisation of the monocular error distribution, we calculate the error in the tangent plane around the point $\mathbf{t}_{\text{true}}$:

$$\Delta\mathbf{t} = \phi_{\mathbf{t}_{\text{true}}}(\mathbf{t}^*). \tag{4.29}$$

Here, $\phi_{\mathbf{t}_{\text{true}}}$ is a orthogonal projection which maps points on the ball with radius $|\mathbf{t}_{\text{true}}|$ onto the tangent plane around $\mathbf{t}_{\text{true}}$ (so that $\mathbf{t}_{\text{true}}$ is mapped to $(0,0)^\top$).

We use two ways to describe the error distribution. Our first measure is based on information theory. As in our preliminary experiment (see Section 4.3.2), we analyse the influence of different parameters $\langle M, N\rangle$ in terms of entropy reduction. Therefore, for each setting $\langle M, N\rangle$ we estimate the sample covariance matrix $\Sigma_{\langle M,N\rangle}$ of the translation error distribution $\Delta\mathbf{t}$. Then, we can compute the entropy reduction in bits,

$$E = \frac{1}{2}\log_2\left(\frac{\det(\Sigma_{\langle M_{\min},15\rangle})}{\det(\Sigma_{\langle M,N\rangle})}\right), \tag{4.30}$$

in relation to the least accurate case where only the minimal number of frames $M_{\min}$[4] and 15 points are used for SLAM.

[4]This is $M_{\min} = 1$ for Setting (i) and $M_{\min} = 2$ for Settings (ii,iii), and $M_{\min} = 4$ for Setting (iv).

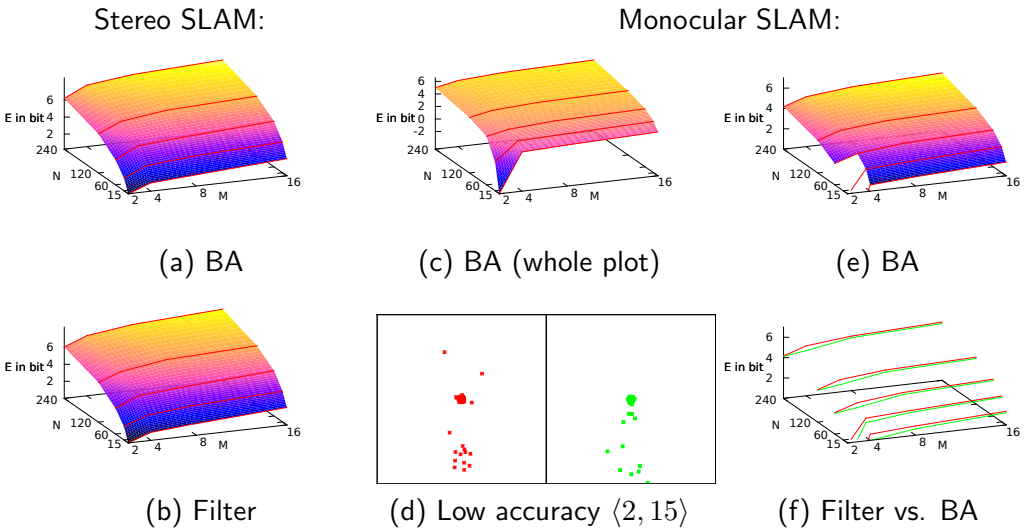


Figure 4.7: **Setting (ii)**. Accuracy plots in terms of entropy reduction in bits (a-c,e,f). Plot (d) illustrates the error distribution for the low robustness case $\langle M, N \rangle = \langle 2, 15 \rangle$. For both BA (left, red) and filtering (right, green), the distributions for this lowest accuracy case contain outliers, i.e. complete SLAM estimation failures, and this explains the discontinuities in the otherwise smooth plots (c,e,f) in the low accuracy corner. Even though we show a range of one metre, a significant portion of outliers lies outside this range.

The influence of the parameters $\langle M, N \rangle$ in Setting (i) is illustrated in Figure 4.6(a-d). As can be seen in all plots (Monocular vs. Stereo, Filtering vs. BA), increasing the number of features leads to a significant entropy reduction. On the other hand, increasing the number of intermediate frames has only a minor influence. Thus, we could reproduce the important result of our preliminary experiment. Also, we can see that the accuracy of our filter is in fact very close to the accuracy of BA, confirming that we have chosen the filter parametrisation well.

The accuracy results for Setting (ii), where the camera still moves sideways but now over a distance such that there is hardly any scene overlap between the first and last frames, are shown in Figure 4.7(a,b). The plots for stereo SLAM look similar to Setting (i). The whole accuracy plot for monocular BA is shown in Figure 4.7(c). Note that for the low accuracy cases $\langle 2, 15 \rangle$ and $\langle 2, 30 \rangle$ the estimation is not very robust, and SLAM fails occasionally. Thus, the corresponding error distributions are heavy tailed/non-Gaussian as shown in Figure 4.7(d), and therefore the entropy reduction measure is not fully meaningful. Therefore, we excluded these two cases from the subsequent analysis and defined $\langle 4, 15 \rangle$ as the minimal base case.

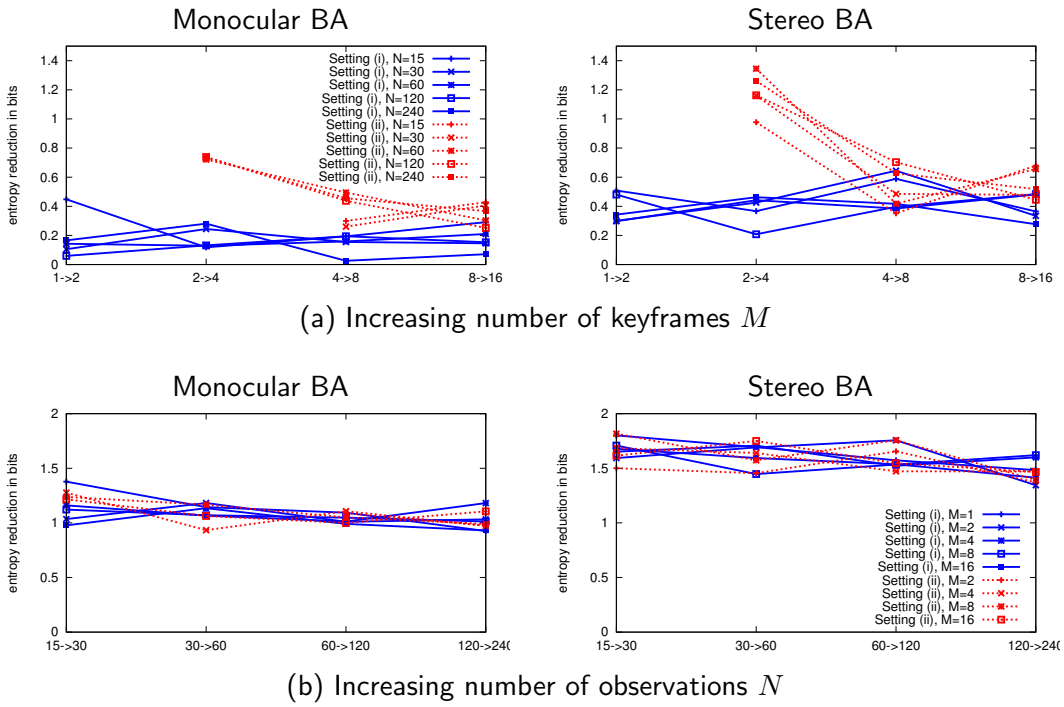


Figure 4.8: Relative entropy reduction when (a) we double the number of intermediate frames and (b) we double the number of observations. Note the difference between Setting (i) (blue, connected lines) versus Setting (ii) (red, dotted lines).

A corresponding accuracy plot is shown in Figure 4.7(e). The characteristic pattern we saw before is repeated: increasing the number of points is the most significant way to increase accuracy. Meanwhile, increasing the number of frames has the main effect of increasing robustness — i.e. avoiding complete failures. Once robustness is achieved, a further increase in M has only a minor effect on accuracy. Finally, as we can see in Figure 4.7(f), monocular BA leads to marginally better accuracy than filtering, especially for small M.

In general, the accuracy plots for Setting (i) and Setting (ii) show a similar pattern. However, there is a significant difference between Setting (i) and Setting (ii). Let us consider the relative entropy reduction when we double the number of intermediate frames, i.e. comparing $\Sigma_{\langle M,N\rangle}$ with $\Sigma_{\langle 2M,N\rangle}$. From Figure 4.8(a), one can clearly see that Setting (ii) benefits more from the increased number of keyframes than Setting (i). This effect is especially prominent for monocular SLAM. While all points are visible in all frames in Setting (i), the scene overlap is larger for more closely placed keyframes in Setting (ii). Increasing the number of observations per frame has a

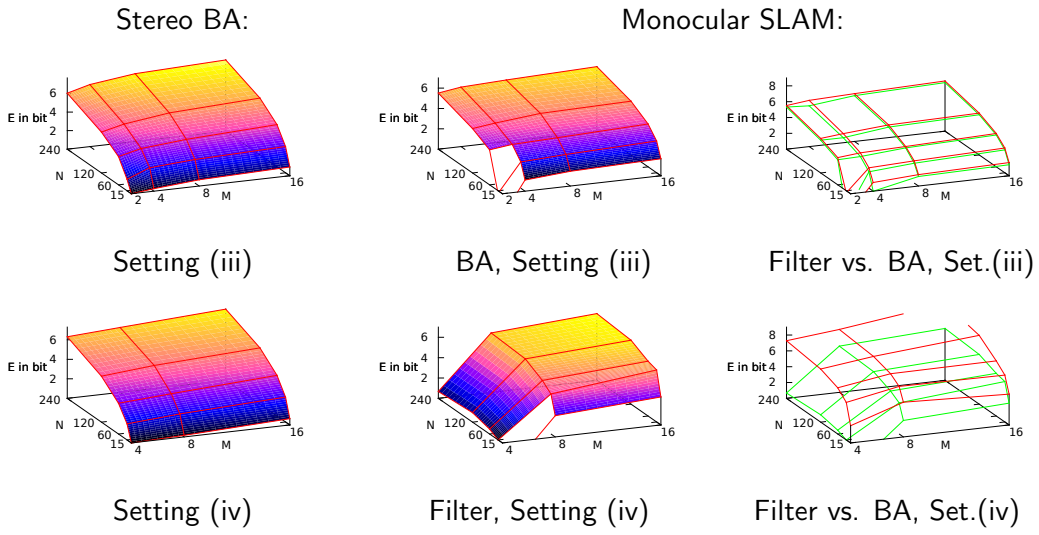


Figure 4.9: Accuracy plots in terms of entropy reduction in bits. The stereo filter leads to very similar results to stereo BA and is therefore not shown here.

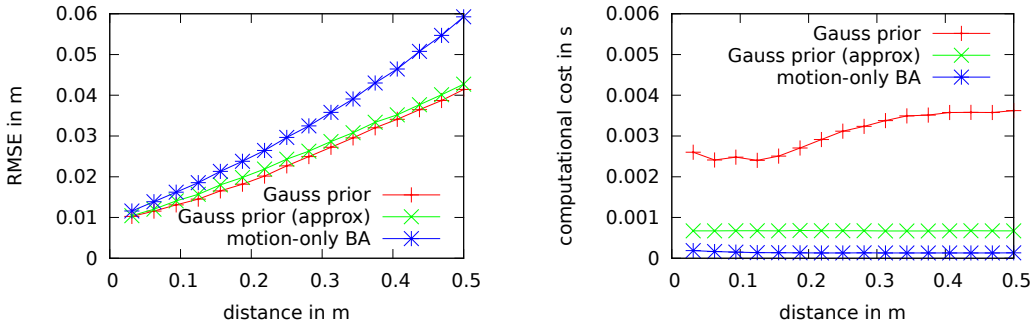
similar impact on both settings (Figure 4.8(b)).

The second error measure we use is the root mean square error (RMSE):

$$R = \sqrt{\frac{1}{k}\sum_{k=0}^{k} \Delta\mathbf{t}_k^2} \tag{4.31}$$

where $k = 500$ is the number of Monte Carlo trials. Compared to the entropy reduction, this is a measure which is not relative but absolute. It is still meaningful for non-Gaussian and non-zero-centred error distributions. The RMSE for Setting (i) is illustrated in Figure 4.6(e,f). In the case that the error distributions are zero-mean Gaussians, entropy reduction and RMSE behave very similarly: they are anti-monotonic to each other. Our main reason for including the entropy reduction is to make our analysis comparable to our preliminary experiment, in which the experiments were performed using covariance propagation and other error metrics such as RMSE were not applicable. We will introduce two combined cost error measures, one relying on entropy reduction and the other on RMSE.

Accuracy plots for the two motion cases with rotational components, Setting (iii,iv), are shown in Figure 4.9. One can see that the result of Setting (iii) is comparable to Setting (ii). This is not surprising since both settings lead to a similar amount of scene overlap. Again, the two low accuracy cases $\langle 2, 15\rangle$ and $\langle 2, 30\rangle$ lead to unstable results and are excluded. For both rotational cases, Setting (iii) and Setting (iv), the



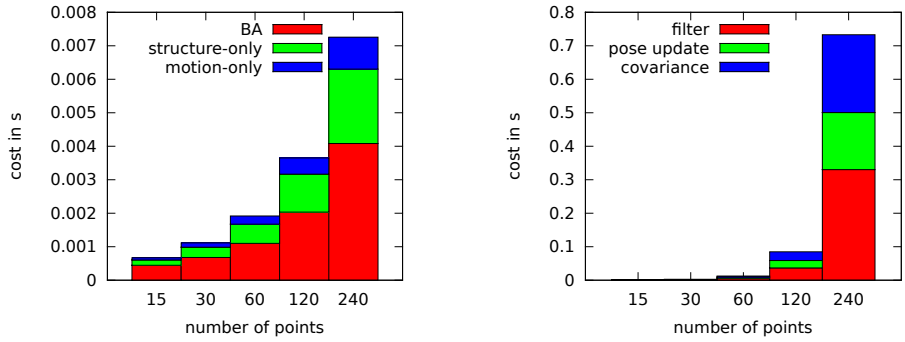
(a) error (b) cost

Figure 4.10: Pose update given known map estimated by monocular filter.

stereo filter approaches the accuracy of stereo BA. However, for the difficult case, monocular vision in Setting (iv), the results are different. BA leads to significantly better results than filtering. Especially for a low number of frames, the performance of the filter is worse. We only removed the very inaccurate case $\langle 2, 15 \rangle$, since it is not practical to exclude all non-robust cases. Even for many features and frames, e.g. $\langle 16, 240 \rangle$, the error distributions are slightly heavy tailed. This low level of robustness might also explain the slightly chaotic, non-monotonic behaviour of the accuracy plots. Thus, conclusions drawn from Setting (iv) have to be considered with care.

4.6.3 The Cost and Accuracy of Motion-Only Estimation for Filter-SLAM

As described in Section 4.5.2, when performing filter-SLAM there are two main options to perform motion-only estimation. Either one can do motion-only BA by minimising equation (4.15) or one can also consider the map uncertainty. While motion-only BA is linear in the number of points N, pose estimation using a Gaussian map prior is cubic in N due to the inversion of innovation matrix $\mathtt{S}$. In a approximated but much more efficient version of this algorithm, the innovation matrix $\mathtt{S}$ and its inverse are only calculated once. For stereo SLAM, we can usually measure the 3D points precisely so that motion-only BA leads to accurate results. However, for a monocular filter where the point depth is uncertain, it is beneficial to consider this uncertainty explicitly (Figure 4.10(a)). Considering map uncertainty in pose estimation leads to a significant increase in computation time (Figure 4.10(b)).



(a) BA-SLAM (b) Filter-SLAM

Figure 4.11: Computational cost of monocular SLAM with respect to the number of points. Note the difference in scale on the vertical cost axis: 0.008 seconds (a) versus 0.8 seconds (b).

In the monocular filtering experiments, we use the approximated version of the algorithm.

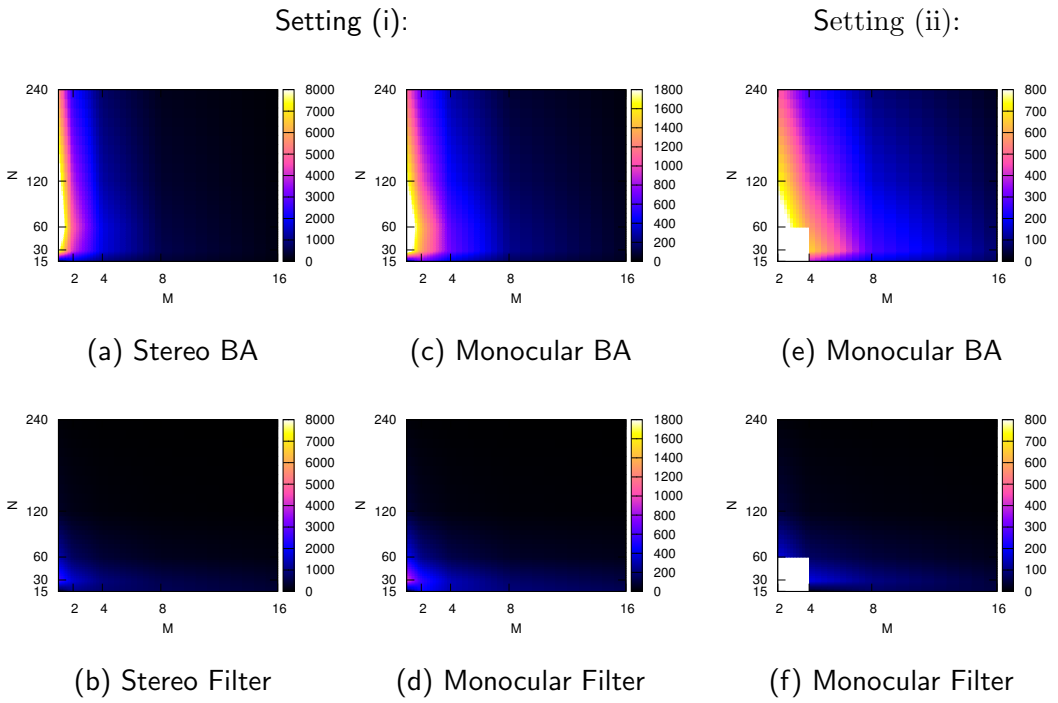
4.6.4 The Cost of Visual SLAM

Under the assumption that all points are visible in all frames, the cost of BA is $O(NM^2+M^3)$, where the first term reflects the Schur complement, while the second term represents the cost of solving the reduced linear system Engels et al. (2006). The costs of structure-only and motion-only estimation are both linear in the number of points. In filtering, the filter update is cubic in the number of observations, which leads to $O(MN^3)$ for the whole trajectory. The cost of pose update given a map is either linear or cubic (see previous section). The cost of the whole SLAM pipelines for varying number of points N are shown in Figure 4.11. Here we illustrate the case of $M = 1$, Setting (i) and monocular SLAM.

4.6.5 Trade-off of Accuracy versus Cost

We would like to analyse the efficiency of BA and filtering for visual SLAM by trading off accuracy against computational cost.[5] First, we do this using the combined

[5]In order to assume the best case for filtering, we do not consider covariance estimation and pose estimation given a known map. Thus, we compare the cost of joint BA updates against the cost of the joint filtering steps for the whole trajectory.



(a) Stereo BA (c) Monocular BA (e) Monocular BA

(b) Stereo Filter (d) Monocular Filter (f) Monocular Filter

Figure 4.12: Accuracy/cost measure in bits per second (bps).

accuracy/cost measure. Thus, we evaluate visual SLAM using entropy by cost in terms of bits per second (bps): $\frac{E}{c}$. E is the amount of entropy reduction as defined in equation (4.30) and c is the average computational cost in seconds of the whole SLAM pipeline. Corresponding plots are shown in Figure 4.12. First one can see that BA seems to be in general more efficient than filtering. Furthermore, there is a pattern that BA is especially efficient for small M, while filtering is only efficient for low accuracy (small M and small N).

Finally, we contrast error with cost in common plots in Figure 4.13. Each curve shows the error and cost for a constant number of frames M and varying number of observations N. For the lowest number of frames (bold curves), we also show results for $N = 480$. In these plots the bottom left corner is the desired area, where we find the highest accuracy and lowest computational cost. For all four settings, we can observe that BA is clearly superior to filtering. Furthermore, we see that for Setting (i) it is always preferable to choose the lowest number of frames. This is still the case for sideways motion BA with partial scene overlap (Setting (ii-iii)) — except for the monocular, low-robustness cases ($M = 2$, and $N \in \{15, 30\}$) which are not shown in the plots. However, for filtering (Setting (ii-iii)), there is actually

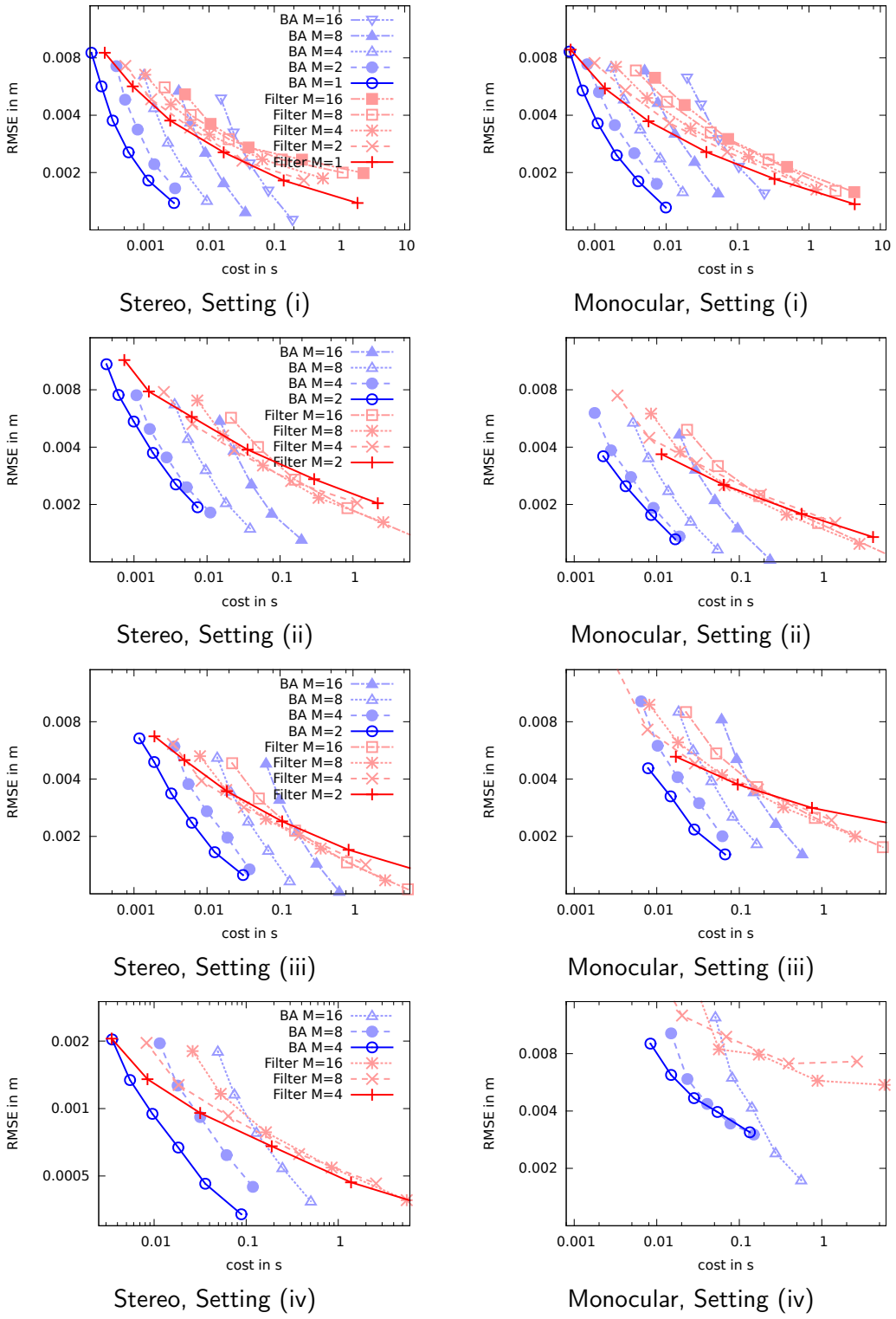


Figure 4.13: Error versus cost on a logarithmic scale.

a cross-over. In order to reach high accuracy, it seems desirable to increase the number of keyframes M. The monocular Setting (iv), low parallax and low scene-overlap, is the most challenging one. The inaccurate case $M = 4$ results in a RMSE greater than $0.02m$ and is therefore not shown. Here, BA outperforms filtering by magnitudes, but increasing M helps the filter. To summarise, it is usually a good strategy to increase the number of points N. Increasing the number of keyframes M seems only to be sensible if both following requirements are fulfilled: First, we use filtering instead of BA, and thus there is significantly higher cost with respect to N compared to M. Second, there is a varying scene overlap which can be maximised with increasing M.

4.7 Discussion

We have shown that filter-SLAM can indeed reach the accuracy of BA for moderately difficult motion patterns and scene structures (Setting (i-iii)), even if we only filter sparse keyframes. In general, increasing the number of points N leads to a significant increase in accuracy, while increasing the number of frames M primarily establishes robustness. Once a level of robustness is reached, a further increase of M has only a minor effect. This shows that the greater efficiency of BA compared to filtering for local SLAM is primarily a cost argument: The cost of BA is linear in N, whereas the cost of filtering is cubic in N. For the sharp forward turn (Setting (iv)) using monocular vision, our analysis is slightly different. It illustrates the known problem of Gaussian filters. Since measurement Jacobians are not re-linearised, the accuracy can significantly decrease compared to BA. Note, however, that the amount of insight we can gain from Setting (iv) is limited. It might be possible to find a better filter parametrisation/implementation which can deal significantly better with this low parallax case. Setting (iv) is merely added as an illustrative example that the accuracy of filtering can be inferior to BA, even for very short trajectories. Here, the dominance of BA compared to filtering is primarily an accuracy argument.

The greater cost of filtering wrt. BA is mainly due to the fact that we represent uncertainties explicitly. In this work, we focused on the SLAM back-end and we did not analyse the accuracy and cost of feature tracking. Instead, we assumed that a perfect data association is given. On one hand, the availability of the covariance can facilitate feature tracking (Neira & Tardós, 2001; Davison, 2005; Chli & Davison,

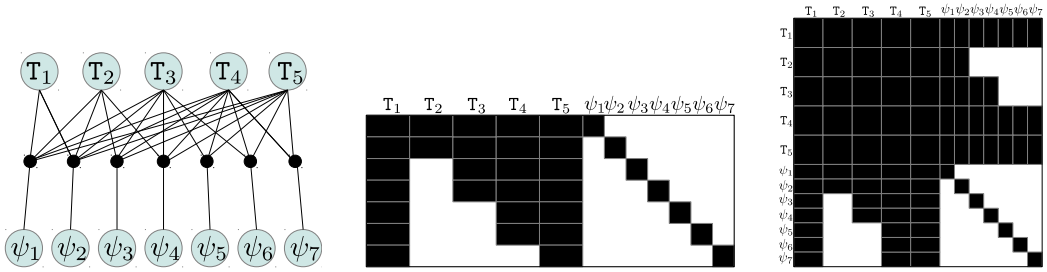
2009; Civera et al., 2009a). On the other hand, modern tracking techniques such as variational optical flow (Werlberger et al., 2010) do not require covariances, and are very effective. For the SLAM back-end, it does not seem beneficial to propagate uncertainties explicitly. Thus, one should only calculate covariances if one needs them elsewhere.

In addition, we did not focus on all aspects of SLAM in our analysis. We intentionally did not consider large-scale SLAM and loop-closing since these issues have been intensively studied in the past. A SLAM framework which works reliably locally, whether it is BA or filtering, can easily be applied to a large scale problems using methods such as sub-mapping or graph-based global optimisation. Furthermore, it was shown that loop-closing can be solved efficiently using appearance-based methods (Nister & Stewenius, 2006; Cummins & Newman, 2009) which can be formulated independently from metric SLAM systems. Thus, we assume in our analysis that the choice between BA and filtering is not relevant at this global level.

4.7.1 Middle Ground between BA and Filtering

While we focussed on the two extreme cases, there is a broad middle ground between filtering and BA.[6] Let us reconsider the three properties of our filter concept defined in Section 4.4. While all three properties are inherently coupled for the EKF, information filters can deal with them independently. Let us lift property 2: Indeed, if we never marginalise out past poses and invisible features, we keep the corresponding information matrix relatively sparse, thus leading to the class of *exactly sparse information filters* (Walter et al., 2007). Figure 4.14(a) shows the corresponding factor graph. In general, each observation connects a point to several poses. However, no point is directly connected to another point. This leads to a similar, but slightly different sparseness structure than standard BA. In BA the factor graph has only binary point-pose constraints $\hat{\mathbf{z}}(\mathtt{T}, \mathbf{y})$ and thus the corresponding Jacobian has one frame block and one point block per row (=observation). The factor graph of this sparse filter has n-ary constraints $\hat{\mathbf{z}}(\mathtt{T}_i, \ldots, \mathtt{T}_{i+n}, \boldsymbol{\psi})$ where $\mathtt{T}_i$ is the anchor frame of $\boldsymbol{\psi}$ and $\mathtt{T}_{i+1}, ..., \mathtt{T}_{i+n}$ are all the following frames in which $\boldsymbol{\psi}$ is visible. Thus, the

[6] Strictly speaking, the Gauss-Newton filter, which we used in the comparison, is already one step towards BA. Some poses — the anchor poses — are not marginalised out; so their means get constantly re-estimated. In addition, the current pose $\mathtt{T}_i$ does not have a motion prior and is therefore 'bundle adjusted'.



(a) Factor Graph (b) Jacobian (c) Hessian

Figure 4.14: Exactly sparse information filter using anchored landmarks.

Jacobian (Figure 4.14(b)) has *several* frame blocks and *one* point block per row. Still, the point block of the Hessian (Figure 4.14(c)) remains block-diagonal, the Schur-complement trick would be applicable, and the algorithmic complexity would decrease to the level of BA. However, there are two caveats. First, if we compute the covariance $\Sigma = \Lambda^{-1}$, the performance benefit would vanish. Thus, we do not have cheap access to the covariance (= forfeit property 3), and therefore lose the main advantage of Gaussian filters. Second, the Jacobians are only linearised once and the update of the information matrix remains additive. Thus, this exactly sparse filter remains inferior to BA.

Another option is to follow the approach of Sibley et al. (2008) and partially lift property 1. One represents some variables using a Gaussian, while others are represented as in BA. In particular, it is sensible to deal with a sliding window of the last current poses using batch processing. All corresponding observations are saved, no uncertainties are maintained and the Jacobians are constantly re-linearised. One represents variables outside this sliding window using a Gaussian distribution, assuming they are well estimated so no further re-linearisation is necessary. This sliding window filter basically performs BA for local motion estimates, and is therefore covered by our analysis.

Typically in BA-SLAM and unlike in filtering, the SLAM problem is solved from scratch each time a new node is added to the graph. Kaess et al. (2008, 2012) introduced a framework for *incremental* BA. For large scale mapping, this framework can have a lower computational cost than batch BA. However, it remains unclear whether there is a significant performance benefit for local SLAM.

4.7.2 Summary

In this chapter, we have presented a detailed analysis of the relative merits of filtering and bundle adjustment for real-time visual SLAM in terms of accuracy and computational cost. We performed a series of experiments using covariance back-propagation and Monte Carlo simulations for motion in local scenes. Starting from a simplified preliminary experiment, we lifted several assumptions by considering partial scene overlap, full SLAM pipelines including monocular bootstrapping, and feature initialisation. Our conclusion is: In order to increase the accuracy of visual SLAM it is usually more profitable to increase the number of features than the number of frames. This is the key reason why BA is more efficient than filtering for visual SLAM. Although this analysis delivers valuable insight into real-time visual SLAM, there is space for further work. In this analysis we assumed known data association. However, the accuracy of a SLAM back-end such as BA is highly coupled with the performance of the visual front-end — the feature tracker. A detailed analysis of this coupling would be worthwhile.

4.8 Bibliographic Remarks

In his seminal paper, Kalman (1960) did not only present an optimal approach for state estimation of linear dynamic systems, which became known as *Kalman filtering*. He also described the duality between control inputs versus observations as well as covariance matrices versus information matrices and therefore laid the foundation of *information filters* too.

Smith et al. (1987) presented the *stochastic map* which is the first recursive formulation of SLAM where the pose and the map are represented using a joint normal distribution. Especially, they point out that under the assumption of a linear model, Kalman filter-based SLAM leads to optimal estimates, while for non-linear measurements — such as angular observations — reasonable results can be obtained in practise.

In the 1990s, the Extended Kalman Filter (EKF) emerged as the standard approach for SLAM (Leonard & Durrant-Whyte, 1991; Betgé-Brezetz et al., 1996; Castellanos, 1998; Davison, 1998; Newman, 1999). Its capabilities were demon-

strated in various simulated and real-robot experiments, but also its drawbacks were studied exhaustively. First, the computational cost grows quadratically with the number of landmarks so that it is only applicable for small scale mapping. Various strategies and filter variations were suggested to overcome this limitation (see Section 5.6). Second, the linearisation of non-linear functions (such as angular measurements) might lead to inconsistencies. Julier & Uhlmann (2001) argued that EKF-SLAM under certain conditions is doomed to diverge. Especially, they showed that the pose covariance of a stationary vehicle, hence using a state prediction with zero covariance, will be reduced during the measurement updates; this results in an overconfident state estimate eventually. Castellanos et al. (2004) showed that inconsistency are coupled with the filter uncertainty. They illustrate that the filter becomes inconsistent earlier when an initial pose uncertainty is assumed. Consequently, they suggested a *robo-centric* parametrisation in order to keep the filter uncertainty small.

In the 2000s, most visual SLAM approaches relied on Gaussian filters. Davison's MonoSLAM (2003) employed the standard EKF and a constant velocity motion model. Jung & Lacroix (2003) performed stereo SLAM on a blimp combining visual odometry and Kalman filtering. Montiel et al. (2006) suggested to use an *inverse depth* parametrisation for monocular SLAM. Especially, they showed that the distribution over uncertain depth of newly initialised points is highly non-Gaussian. On the other hand, inverse depth can be well represented by a normal distribution. An inverse depth feature representation leads to a more linear observation model and therefore more accurate and consistent results. Pietzsch (2008) presented an improved anchored inverse depth parametrisation where simultaneous initialized feature share a common anchor frame. This reduces the dimension of the filter state and therefore the computational cost significantly. In their MonoSLAM adaptation, Holmes et al. (2009) compared the EKF with Unscented Kalman Filtering (UKF, Julier & Uhlmann, 2004) — an improved non-linear filter variant with sigma point linearisation. In particular, they incorporated an efficient square-root implementation. In a set of experiments, Holmes et al. showed that UKF-based monocular SLAM leads to more consistent estimates, but its computational cost is more than ten times higher. They concluded that the "EKF remains properly the algorithm of choice" since the square-root UKF "is outweighed by the speed handicap". Civera et al. (2009b) adapted the approach of Castellanos et al. (2004, 2007) to monocular SLAM and used a *camera-centric* filter parametrisation in order to minimize the

negative impact of non-linearities during exploration. The *Gauss-Newton filter* we use in this chapter can be characterised as an iterative extended information filter with uniform pose priors. The term *Gauss-Newton filter* is borrowed from Sibley et al. (2005) who employed it for long-range stereo. Eade & Drummond (2007) or rather (Eade, 2008, Chap. 6) used the Gauss-Newton filter in their monocular submapping framework, and presented the approach for pose estimation given a map prior which we incorporated in our analysis.

From the field of photogrammetry, Bundle Adjustment (BA) emerged as the gold standard for structure and motion batch estimation. In the 2000s, visual odometry-type frameworks incorporated sliding window BA strategies for efficient incremental motion estimation (see Section 3.7). With their stand-out system PTAM, Klein & Murray (2007) demonstrated that BA can be used for real-time visual SLAM if a keyframe sparsification is used.

Comparisons between Gaussian filtering and least-squares optimisation have been presented in the past. Bell & Cathey (1993) showed that "the iterated Kalman filter (IKF) update is an application of the Gauss-Newton method for approximating a maximum likelihood estimate." In the context of 2D bearing-only SLAM, Deans & Herbert (2001) performed a experimental comparison between BA and Kalman filtering in a loop closure scenario. They argued that EKF-SLAM is computational more efficient in this setting since the state space stays compact, while the state space of BA grows constantly with each new measurement. However, the EKF-SLAM is doomed to diverge — mainly due to the non-linearities of bearing-only landmark initialisation. Deans & Herbert suggest to combine BA and filtering such that some variables are filtered while others are used in a batch approach so that a better linearisation as in the standard EKF is achieved. This approach was taken further by Sibley et al. (2008). Their *sliding window filter* is a hybrid between BA and filtering, and was used for stereo SLAM. In the context of camera calibration and structure from motion, a similar approach was described earlier by McLauchlan & Murray (1996, 1995)

In our analysis, we concentrated on techniques which assume a Gaussian distribution over the joint SLAM state, or, equivalently, which minimizes a quadratic cost. Other parametric and non-parametric approaches were presented for the joint estimation of structure and motion. Hartley & Schaffalitzky (2004) suggested to solve structure from motion problems by minimizing the L_∞-norm instead of the common

quadratic cost functions. This approach has the advantage that it leads to a convex optimisation problem; instead of multiple local minima, the corresponding cost function has a single minimum. On the other hand, such an approach requires that the data is absolutely outlier free. Montemerlo & Thrun (2003) presented FastSLAM where the distribution over the pose is represented by a set of particles. Each particle carries a set of independent landmark representations exploiting the property of Rao-Blackwellization: Two landmarks are *conditionally independent* given the pose is known. Such a Rao-Blackwellized Particle Filter (RBPF) has a space and time complexity which is linear in the number of particles. Visual SLAM approaches using an RBPF include Sim et al.'s stereo framework (2005) and Eade & Drummond's monocular framework (2006). Bailey et al. (2006) argued that FastSLAM performs a non-optimal local search and inconsistent estimates are unavoidable in the long run. These results were confirmed by Eade (2008, pp.139) in the context of monocular SLAM.

This chapter is mainly based on Strasdat et al. (2012) and partially on Strasdat et al. (2010a).

Chapter 5

Scale Drift-Aware Large Scale Monocular SLAM

In which we introduce a new pose-graph optimisation technique which allows for the correction of rotation, translation and scale drift at loop closures.

Accurate and efficient local motion estimation is a crucial component of a real-time SLAM framework. However, structure and motion estimates are prone to drift over time — which is especially apparent if SLAM is applied in a large-scale setting. The main challenge is to maintain a map representation which is globally consistent. Once a known place is revisited, the error in the estimate needs to be propagated over a chain of poses. In stereo vision-based SLAM, there have been systems presented which offer end to end solution for real-time large-scale SLAM such as Konolige & Agrawal (2008) who combined visual odometry (to achieve local accuracy) with pose-graph optimisation (to ensure global consistency). Developing such a large scale SLAM system for monocular vision proved to be more difficult. A classic issue with monocular visual SLAM is that due to the purely projective nature of a single camera, motion estimates and map structure can only be recovered up to scale (see Figure 1.2, p.17). The fact that a single camera does not measure metric scale means that either scale has to be introduced by an additional information source (such as a calibration object as in Davison (2003) or even by exploiting

nonholonomic motion constraints (Scaramuzza et al., 2009)). Or one must proceed mapping with scale as an undetermined factor. Since we do not want to rely on any additional prior information, we follow the latter approach and invent an arbitrary scale at the beginning. Due to the unobservability of absolute scale, however, the scale of locally constructed map portions and the corresponding motion estimates is liable to drift over time — a problem, we have to account for.

Our analysis of filtering versus BA, presented in the previous chapter, indicates that keyframe approaches — with BA at the local level — are strongly advantageous compared to methods whose building blocks are based on filtering. In light of this analysis, the previous large scale monocular systems (Eade & Drummond, 2007; Clemente et al., 2007; Pinies & Tardós, 2008) can be seen as somewhat unsatisfactory approaches, which combine filtering at a local level with optimisation at the global level. For instance, when loop closures are imposed, the relative positions of their filtered submaps changes, but any drift within the local maps themselves cannot be resolved. In this chapter, we therefore present a new pose-graph optimisation technique which allows for the efficient correction of rotation, translation and scale drift at loop closures. In combination with the monocular exploration approach of Chapter 3, it leads to a keyframe optimisation approach from top to bottom, aiming for maximum accuracy while taking into full account the special character of SLAM with monocular vision. Our approach is based on the Lie group of similarity transformations which is discussed in detail. Furthermore, we present a framework for how the Jacobians of general pose-graph optimisation problems can be approximated efficiently. Our approach is proven via large-scale simulation and real-world experiments where a camera completes large looped trajectories.

5.1 Gauge Freedoms, Monocular SLAM and Scale Drift

Metric SLAM systems aim to build coherent maps, in a single coordinate frame, of the areas that a robot moves through. But they must normally do this based on purely relative measurements of the locations of scene entities observable by their on-board sensors. As discussed in Section 3.3.2, there are always certain degrees of *gauge freedom* in the maps that they create, even when the best possible job is done

of estimation. These gauge freedoms are degrees of transformation freedom through which the whole map, consisting of feature and robot position estimates taken together, can be transformed without affecting the values of the sensor measurements. In SLAM performed by a robot moving on a ground plane and equipped with a range-bearing sensor, there are three degrees of gauge freedom, since the location of the entire map with regard to translations and rotation in the plane is undetermined by the sensor measurements. In SLAM by a robot moving in 3D and equipped with a sensor like calibrated stereo vision or a 3D laser range-finder, there are six degrees of gauge freedom, since the whole map could experience a rigid body transformation in 3D space. In monocular SLAM, however, there are fundamentally seven degrees of gauge freedom (Triggs et al., 1999), since the overall scale of the map, as well as a 6 DoF (degrees of freedom) rigid transformation, is undetermined (scale and a rigid transformation taken together are often known as a similarity transformation).

It is the number of gauge degrees of freedom in a particular type of SLAM which therefore determines the ways in which drift will inevitably occur between different fragments of a map. Consider two distant fragments in a large map built continuously by a single robot: local measurements in each of the fragments have no effect on pulling either towards a particular location in the degrees of gauge freedom. If they are not too distant from each other, they will share some coherence in these degrees of freedom, but only via compounded local measurements along the chain of fragments connecting them. The amount of drift in each of these degrees of freedom will grow depending on the distance between the fragments, and the distribution of the potential drift can be calculated if needed by uncertainty propagation along the chain.

It is very well known that planar maps built by a ground-based robot drift in three degrees of freedom. Furthermore maps built by 3D range-bearing sensors such as stereo cameras drift in six degree of freedom; so maps built by a monocular camera with no additional information drift in seven degrees of freedom. It is through these degrees of freedom therefore which loop closure optimisations must adjust local map fragments (poses or submaps in a graph).

5.2 The Group of Similarity Transformations

Our approach is based on the group of rotation, translation and scaling in 3D, in other word the group of *similarity transformations* which we will introduce now.

Let us first consider the group of rotation and scaling which consists of matrices of the following form:

$$s\mathtt{R} \quad \text{with} \quad s \in \mathbb{R}^+ \quad \text{and} \quad \mathtt{R} \in \mathbf{SO}(n) \tag{5.1}$$

The group of rotations and scaling can be seen as the direct product $\mathbb{R}^+(n) \times \mathbf{SO}(n)$ of special orthogonal group $\mathbf{SO}(n)$ and the group of scaling $\mathbb{R}^+(n)$ (Section 2.4.9). A matrix $\mathtt{A} \in \mathbb{R}^+(n) \times \mathbf{SO}(n)$ has the following properties: $\mathtt{A}\mathtt{A}^\top = \mathtt{A}^\top\mathtt{A} = s^2\mathtt{I}$ and $\det(\mathtt{A}) = s^n$. Thus, $\mathbb{R}^+(n) \times \mathbf{SO}(n)$ is a proper generalisation of the well-studied special orthogonal group $\mathbf{SO}(n)$. Let us now concentrate on the three dimensional case. The tangent space of $\mathbb{R}^+(3) \times \mathbf{SO}(3)$ consists of matrices of the form $\mathtt{X} = [\boldsymbol{\omega}]_\times + \sigma\mathtt{I}_{3\times 3}$. Since rotation and scaling commute, one can verify easily that the exponential map is

$$\exp([\boldsymbol{\omega}]_\times + \sigma\mathtt{I}]) = e^\sigma \exp([\boldsymbol{\omega}]_\times) \ . \tag{5.2}$$

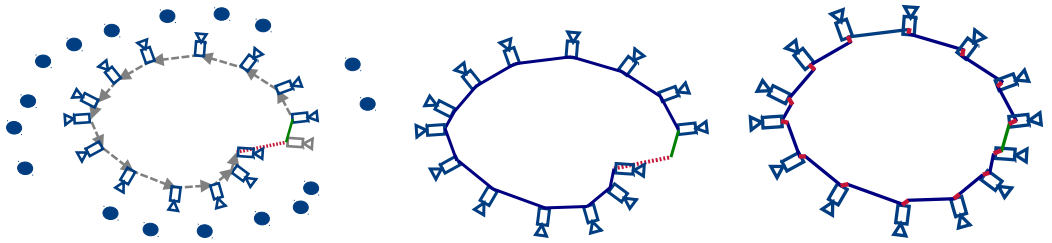
The exponential map $\exp_{\mathbb{R}^+(3)}$ is surjective. This follows from the surjectivity of $e : \mathbb{R} \to \mathbb{R}^+$ and $\exp : \mathfrak{so}(3) \to \mathbf{SO}(3)$ and thus the logarithm can be calculated as

$$\log(s\mathtt{R}) = \log(\mathtt{R}) + \ln(s)\mathtt{I} \ . \tag{5.3}$$

The purpose of the following side-note is to underline the close relation between $\mathbf{SO}(3)$ and $\mathbb{R}^+(3) \times \mathbf{SO}(3)$: It is commonly known that the group of unit quaternions is homomorph to the group of rotation $\mathbf{SO}(3)$. In particular, $\mathbf{SU}(2)$ is the *double cover* of $\mathbf{SO}(3)$ so that each rotation matrix has two quaternion representations. Analogously, the group of non-zero quaternions is the double cover of the group of rotation and scaling, so that quaternions are an elegant way to represent elements of $\mathbb{R}^+(3) \times \mathbf{SO}(3)$.

Now, we can define the group of similarity transformations which is a generalisation of $\mathbf{SE}(3)$ by including a scale factor s. Thus we have matrices $\mathtt{S}$ of the form

$$\mathtt{S} = \begin{pmatrix} s\mathtt{R} & \mathbf{t} \\ \mathtt{0}_{1\times 3} & 1 \end{pmatrix} \quad \text{with} \quad s\mathtt{R} \in \mathbb{R}^+(3) \times \mathbf{SO}(3) \quad \text{and} \quad \mathbf{t} \in \mathbb{R}^3 \ . \tag{5.4}$$



(a) Poses and points (b) Pose-graph (c) Corrected pose-graph

Figure 5.1: The loop closure problem. (a) illustrates the error (dotted red line) between the final camera pose (in grey) and the drifted pose estimate. The loop closure constraint is shown in green. (b) illustrates the pose-graph representation, where point observations are replaced by relative pose graph constraints (blue line). We correct the pose graph (c) and close the loop by distributing the error over all relative constraints.

Analogous to the tangent space of $\mathbf{SE}(3)$, members of the tangent space $\mathfrak{sim}(3)$ are of the form

$$\mathtt{Y} = \begin{pmatrix} [\boldsymbol{\omega}]_\times + \sigma \mathtt{I}_{3\times 3} & \boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 0 \end{pmatrix} \quad \text{with} \quad \boldsymbol{\upsilon} \in \mathbb{R}^3 \,, \quad \boldsymbol{\omega} \in \mathbb{R}^3 \quad \text{and} \quad \sigma \in \mathbb{R}. \tag{5.5}$$

Furthermore, we show in Appendix A.5 that the exponential map $\exp : \mathfrak{sim}(3) \to \mathbf{Sim}(3)$ has the following closed form expression:

$$\exp_{\mathbf{Sim}(3)}(\boldsymbol{\upsilon}, \boldsymbol{\omega}, \sigma) := \exp \begin{pmatrix} [\boldsymbol{\omega}]_\times + \sigma \mathtt{I}_{3\times 3} & \boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 0 \end{pmatrix} = \begin{pmatrix} e^\sigma \exp([\boldsymbol{\omega}]_\times) & \mathtt{W}\boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 1 \end{pmatrix} \tag{5.6}$$

with

$$\mathtt{W} = \left(\frac{e^\sigma - 1}{\sigma}\right)\mathtt{I} + \frac{A\sigma + (1-B)\theta}{\theta(\sigma^2+\theta^2)}[\boldsymbol{\omega}]_\times + \left(\frac{e^\sigma - 1}{\sigma} - \frac{(B-1)\sigma + A\theta}{\sigma^2+\theta^2}\right)\frac{[\boldsymbol{\omega}]_\times^2}{\theta^2} \,, \tag{5.7}$$

$A = e^\sigma \sin(\theta)$, $B = e^\sigma \cos(\theta)$ and $\theta = ||\boldsymbol{\omega}||_2$. The corresponding logarithm is

$$\log \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathtt{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} \log(\mathtt{R}) + \ln(s)\mathtt{I} & \mathtt{W}^{-1}\mathbf{t} \\ \mathtt{0}_{1\times 3} & 0 \end{bmatrix} . \tag{5.8}$$

5.3 Loop Closure

Let us consider a loop closure scenario in a large-scale map (as exemplified by Figure 5.1(a)). The camera is travelling around in a cycle and returns close to its start position. However, because of drift, there is an error between the final pose

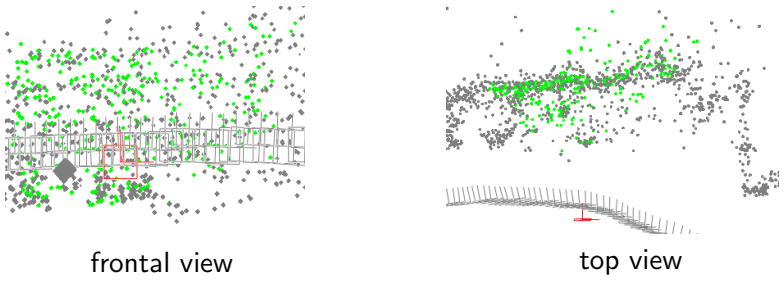


Figure 5.2: Illustration of depth estimation using nearest neighbour interpolation. 3D map points are shown in grey, centres of SURF with interpolated depth are shown in green.

and its estimate. The loop closure problem is typically divided into two sub-tasks: First we detected that a place is revisited. This ultimately involves the detection of a geometric constraint between two camera views which typically belong to two topologically distant fragments of the map. Second, we correct the drift in the final pose estimate by distributing the error along the chain of camera pose constraints.

5.3.1 Loop Closure Detection

It is well known that loop closures can be detected effectively using appearance information only (Nister & Stewenius, 2006; Angeli et al., 2008; Cummins & Newman, 2009). These methods often rely on visual bags of words based on SIFT or SURF features. Given that we have a loop closure detected between two frames associated with a set of feature matches, the standard method would apply RanSaC in conjunction with the 5-point method (Nistér, 2004) in order to estimate the epipolar geometry. Then the relative Euclidean motion up to an unknown scale in translation can be recovered.

However, we can exploit the fact that in our SLAM system each frame is associated with a large set of three-dimensional feature points. First, we create a candidate set of SURF feature pairs by matching features between the current frame and the loop frame based on their descriptors. Then, we create a dense surface model using the three-dimensional feature points visible in both frames. Next, the unknown depths of the SURF features of the loop frame are calculated using this dense surface model.[1]

[1]The underlying assumption is that the scene structure is smooth enough. However, this assumption is not only vital if dense surface models are used, but always if we aim to calculate the

The computationally very efficient k-nearest neighbour regression algorithm proved to be sufficient for our needs. In other words, we calculate the depth of SURF features by simply interpolating the depth of nearby 3D points (Figure 5.2). Note that there are more sophisticated, but still computationally efficient dense surface models available such as implicit surfaces (Newcombe & Davison, 2010). Finally, a 7 DoF similarity constraint $\mathtt{S}_{\text{loop}}$ can be calculated based on the 3D-3D SURF correspondences in a three point RanSaC scheme. In particular given three such correspondences $\mathbf{x}_i \leftrightarrow \bar{\mathbf{x}}_i$, one can calculate the similarity transformation

$$\mathtt{S} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathtt{0} & 1 \end{bmatrix} \quad \text{with} \quad \mathbf{x}_i = s\mathtt{R}\bar{\mathbf{x}}_i + \mathbf{t} \quad \text{for} \quad i \in 1,2,3 \tag{5.9}$$

$$\tag{5.10}$$

in a closed form approach following Arun et al. (1987):

$$\mathbf{c} = \frac{1}{3}(\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3), \quad \bar{\mathbf{c}} = \frac{1}{3}(\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2 + \bar{\mathbf{x}}_3), \quad \text{(calculate centroids)} \tag{5.11}$$

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{c}, \quad \bar{\mathbf{y}}_i = \bar{\mathbf{x}}_i - \bar{\mathbf{c}}, \quad \text{(subtract centroids from points)} \tag{5.12}$$

$$\mathtt{H} = \mathbf{y}_1\bar{\mathbf{y}}_1^\top + \mathbf{y}_2\bar{\mathbf{y}}_2^\top + \mathbf{y}_3\bar{\mathbf{y}}_3^\top, \quad \mathtt{U}\cdot\Sigma\cdot\mathtt{V}^\top = \mathtt{H} \quad \text{(singular value decomposition)} \tag{5.13}$$

$$\mathtt{R} = \mathtt{V}\mathtt{U}^\top, \quad s = \frac{\sqrt{||\mathbf{y}_1||_2^2 + ||\mathbf{y}_2||_2^2 + ||\mathbf{y}_3||_2^2}}{\sqrt{|\bar{\mathbf{y}}_1||_2^2 + ||\bar{\mathbf{y}}_2||_2^2 + ||\bar{\mathbf{y}}_3||_2^2}}, \quad \mathbf{t} = \mathbf{c} - s\mathtt{R}\bar{\mathbf{c}} \tag{5.14}$$

Given this initial transformation estimate, more matches can be found using guided search, and the transformation is refined using robust optimisation (i.e. bundle adjustment using a robust kernel).

5.3.2 Loop Closure Correction

After the loop closure is detected and feature matches are found between the two frames, the loop closing problem can be stated as a large BA problem. We have to optimize over all frames and points in the cycle (Figure 5.1(a), p.127). However, optimising over a large number of frames is computationally demanding. More seriously, since BA is not a convex problem, and we are far away from the global minimum, it is possible that BA will get stuck in a local minimum.

One solution is to optimise over relative constraints between poses using *pose-graph optimisation* (Lu & Milios, 1997; Agrawal, 2006). In a first step one usually

3D position of a blob feature no matter which method is used. In other words, the 3D position of a blob is only properly defined if its 'carrying' surface is smooth enough.

marginalises out the points in a way such that a set of point measurements between two frames is replaced by a single relative pose constraint T_{ji}. This leads to a pose graph as shown in Figure 5.1(b) on page 127. Let us consider two absolute poses T_i and T_j.[2] The relative constraint between two such initial pose estimates are calculated as: $\mathsf{T}_{ji} := \mathsf{T}_j \cdot \mathsf{T}_i^{-1}$, except for the loop closure constraint which is calculated as described above. These relative pose constraints T_{ji} are now regarded as measurements (*constants*). The target of the optimisation is to modify the absolute poses T_j, T_i in a way such that the pose concatenations $\mathsf{T}_{ji} \cdot \mathsf{T}_i \cdot \mathsf{T}_j^{-1}$ are as close to the identity as possible. Initially, all pose concatenations equals the identity except for the one containing the loop closure constraint (Figure 5.1(b), p.127). The purpose of the pose graph optimisation is to distribute this error over all constraints, and hence close the loop as illustrated in Figure 5.1(c).

We define the residual error $\mathbf{d}_{i,j}$ between two poses in the tangent space

$$\mathbf{d}_{i,j} := \log(\mathsf{T}_{ji} \cdot \mathsf{T}_i \cdot \mathsf{T}_j^{-1})^\vee \ , \tag{5.15}$$

where $\log(\cdot)^\vee := (\log(\cdot))^\vee$ is the logarithmic map plus vee-operator which maps elements of the Lie group (e.g. $\mathbf{SE}(3)$) to the minimal tangent vector representation (e.g. $\mathbb{R}^6$). Specifying the error in the tangent space can be done safely for all groups $\mathbf{G}$ whose exponential map is surjective, which is true for all examples we have considered so far; this ensures that the logarithm is defined for all $\mathsf{T}_k \in \mathbf{G}$.[3] If the exponential map is not one-to-one, we have to make sure that the image of the logarithm is defined around zero. In particular, if the underlying Lie group is $\mathbf{SO}(3)$ or any from $\mathbf{SO}(3)$ derived group such as $\mathbf{SE}(3)$ or $\mathbf{Sim}(3)$, it is important to define the logarithm in such a way that it returns elements with $\theta = ||\boldsymbol{\omega}||_2 \in [0, \pi]$, and not for instance $\theta \in [0, 2\pi]$; this way we make sure that group members close to the identity are mapped to tangent vectors close to zero. Similarly, if the underlying group is $\mathbf{SO}(2)$, the in-plane rotation angle should lie in $[-\pi, \pi]$.

We can formulate pose-graph optimisation as a least-squares problem by minimising the energy

$$\chi^2(\mathsf{T}_2, ..., \mathsf{T}_m) := \sum_{\mathsf{T}_{ji}} \mathbf{d}_{i,j}^\top \Lambda_{\mathsf{T}_{ji}} \mathbf{d}_{i,j} \ , \tag{5.16}$$

[2] To be more precise, we have two poses T_{iw} and T_{jw} where w is the global world reference frame. In the following the w is dropped, but we keep in mind that absolute poses denote point transformations from the world reference frame w into the particular camera reference frame.

[3] If however, the exponential map is not surjective, such as for the special linear group $\mathbf{SL}(3)$ which is used to represent homographies (Mei et al., 2008), one has to make sure that the relative errors $\mathsf{T}_{ji} \cdot \mathsf{T}_i \cdot \mathsf{T}_j^{-1}$ are small enough so that they fall into the domain of the matrix logarithm.

with respect to the absolute poses $\mathtt{T}_2, ..., \mathtt{T}_m$. The first transform $\mathtt{T}_1$ is typically fixed and defines the coordinate frame. The inverse covariance $\Lambda_{\mathtt{T}_{ji}}$ of the relative pose constraint is often simply set to the identity, but can be computed accurately using point marginalisation and lifting (Konolige & Agrawal, 2008).

In the BA formulation, scale is an implicit parameter. For instance, one could understand the scale linked to a particular pose $\mathtt{T}_k$ as the average scene depth; the average depth of all points visible in pose $\mathtt{T}_k$. However, since all points are eliminated in the pose graph formulation, these implicit parameters vanish. Thus, if we were to perform an optimisation using 6 DoF rigid body transformation $\mathtt{T}_k \in \mathbf{SE}(3)$, we can efficiently correct translation and rotational drift. However, it would not deal with scale drift, and would lead to an unsatisfactory overall result as we also will confirm experimentally in Section 5.4. Therefore, we perform optimisation based on 7 DoF similarity constraints $\mathtt{S}_k \in \mathbf{Sim}(3)$ where the scale of a particular pose is represented explicitly. In order to prepare for the 7 DoF optimisation, we transform each absolute pose $\mathtt{T}_k$ to an absolute similarity $\mathtt{S}_k$, and each relative pose constraint $\mathtt{T}_{ji}$ to a relative similarity constraint $\mathtt{S}_{ji}$ by leaving rotation and translation unchanged and setting the scale $s = 1$. Only the relative loop constraint $\mathtt{S}_{\text{loop}}$ has a scale $s_{\text{loop}} \neq 1$ (as explained in the previous section). Thus, the residual $\mathbf{d}_{i,j}$ between two transformations $\mathtt{S}_i$ and $\mathtt{S}_j$ minimally in the tangent space $\mathfrak{sim}(3)$ is:

$$\mathbf{d}_{i,j} = \log(\mathtt{S}_{ji} \cdot \mathtt{S}_i \cdot \mathtt{S}_j^{-1})^{\vee}_{\mathfrak{sim}(3)} \ . \tag{5.17}$$

We solve the corresponding least squares problem (5.16) using Levenberg Marquardt. Its Jacobian is sparse with two dense blocks per constraint/row. We exploit the sparseness pattern using sparse Cholesky (Davis, 2006).

After the similarities $\mathtt{S}_i^{\text{cor}}$ are corrected, we also need to correct the set of points. For each point $\mathbf{y}_j$, a frame $\mathtt{T}_i$ is selected in which it is visible. Now we can map each point relative to its corrected frame: $\mathbf{y}_j^{\text{cor}} = (\mathtt{S}_i^{\text{cor}})^{-1}(\mathtt{T}_i \mathbf{y}_j)$. Afterwards, each similarity transform $\mathtt{S}_i^{\text{cor}}$ is transformed back to a rigid body transform $\mathtt{T}_i^{\text{cor}}$ by setting the translation to $s\mathbf{t}$ and leaving the rotation unchanged. Finally, the whole map can be further optimised using structure-only or full BA.

5.3.3 Jacobian of Pose-graph Optimisation

Let $\mathtt{T}_i, \mathtt{T}_j, \mathtt{T}_{ji}$ being elements of a respective Lie group (e.g. $\mathbf{SE}(3)$, or $\mathbf{Sim}(3)$). In order to perform pose graph optimisation in a least square manner, we need to calculate the Jacobians

$$\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{T}_{ji} \exp(\widehat{\boldsymbol{\epsilon}}) \mathtt{T}_i \mathtt{T}_j^{-1})^\vee|_{\boldsymbol{\epsilon}=\mathbf{0}} \tag{5.18}$$

and

$$\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{T}_{ji} \mathtt{T}_i (\exp(\widehat{\boldsymbol{\epsilon}}) \mathtt{T}_j)^{-1})|_{\boldsymbol{\epsilon}=\mathbf{0}} \overset{(2.73)}{=} -\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\exp(\widehat{\boldsymbol{\epsilon}}) \mathtt{T}_j \mathtt{T}_i^{-1} \mathtt{T}_{ji}^{-1})^\vee|_{\boldsymbol{\epsilon}=\mathbf{0}} \,. \tag{5.19}$$

We can treat both Jacobians in a unified manner by considering

$$\mathtt{J}_{\mathtt{A}}^{\mathtt{B}} := \frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{A} \exp(\boldsymbol{\epsilon}) \mathtt{B})^\vee|_{\boldsymbol{\epsilon}=\mathbf{0}} \tag{5.20}$$

and setting $\mathtt{A} := \mathtt{T}_{ji}, \mathtt{B} := \mathtt{T}_i \mathtt{T}_j^{-1}$ or $\mathtt{A} := \mathtt{I}, \mathtt{B} := \mathtt{T}_j \mathtt{T}_i^{-1} \mathtt{T}_{ji}^{-1}$ respectively. Using the chain rule, the Jacobian $\mathtt{J}_{\mathtt{A}}^{\mathtt{B}}$ could be calculated under the assumption that the derivative of the matrix logarithm $\frac{\partial}{\partial \mathtt{X}_{i,j}} \log(\mathtt{X})^\vee$ would be known. However, unlike the matrix exponential whose derivatives can be calculated efficiently and represented compactly,

$$\frac{\partial \exp(\widehat{\boldsymbol{\epsilon}}) \mathtt{T}}{\partial \boldsymbol{\epsilon}_i}|_{\boldsymbol{\epsilon}=0} = \mathtt{G}_i \cdot \mathtt{T} \quad \text{(restating equation (2.102))} \,,$$

no such trick exists for the matrix logarithm. A symbolic expression for $\frac{\partial}{\partial \mathtt{X}_{i,j}} \log(\mathtt{X})^\vee$ can be only derived for particular Lie groups $\mathbf{G}$ where a closed form expression of $\log : \mathbf{G} \to \mathfrak{g}$ exists. Even if such a closed form expression of the logarithm is known (as for $\mathbf{SO}(3)$), it is typically not very compact so that the symbolic expression of the derivative $\frac{\partial}{\partial \mathtt{X}_{i,j}} \log(\mathtt{X})^\vee$ is rather lengthy; the corresponding symbolic expression for $\mathtt{J}_{\mathtt{A}}^{\mathtt{B}}$ would be even more complicated and hence costly to compute.

Therefore, we suggest the following approach by exploiting that

$$\begin{aligned} \mathtt{J}_{\mathtt{A}}^{\mathtt{B}} &= \frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{A} \exp(\boldsymbol{\epsilon}) \mathtt{B})^\vee \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \overset{(2.66,2.77)}{=} \frac{\partial}{\partial \boldsymbol{\epsilon}} \log\left(\exp(\widehat{\mathtt{Ad}_{\mathtt{A}} \boldsymbol{\epsilon}}) \mathtt{AB}\right)^\vee \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \\ &= \frac{\partial}{\partial \boldsymbol{\epsilon}} \log\left(\exp(\widehat{\mathtt{Ad}_{\mathtt{A}} \boldsymbol{\epsilon}}) \exp(\widehat{\mathbf{d}})\right)^\vee \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} = \frac{\partial}{\partial \boldsymbol{\epsilon}} \mathrm{cbh}(\mathtt{Ad}_{\mathtt{A}} \boldsymbol{\epsilon}, \mathbf{d}) \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \,. \end{aligned} \tag{5.21}$$

Here $\mathtt{Ad}_{\mathtt{A}}$ is the adjoint of $\mathtt{A}$, $\mathbf{d} := \log(\mathtt{AB})^\vee$, and $\mathrm{cbh}(\cdot, \cdot) = \log(\exp(\widehat{\mathbf{x}}) \cdot \exp(\widehat{\mathbf{y}}))$ the Campbell-Baker-Hausdorff formula (described in Appendix A.4). Thus, we can

Group	σ	$\|\|\mathbf{d}\|\|_2$	$\|\|\mathtt{J}_{\text{num}} - \mathtt{J}_1\|\|_2$	$\|\|\mathtt{J}_{\text{num}} - \mathtt{J}_2\|\|_2$	$\|\|\mathtt{J}_{\text{num}} - \mathtt{J}_3\|\|_2$
$\mathbf{SO}(3)$	0.0005	$8.0622 \cdot 10^{-4}$	$5.7008 \cdot 10^{-4}$	$8.7931 \cdot 10^{-8}$	$1.2267 \cdot 10^{-10}$
	0.015	$2.5554 \cdot 10^{-2}$	$1.8070 \cdot 10^{-2}$	$8.5531 \cdot 10^{-5}$	$7.8969 \cdot 10^{-9}$
	0.5	$7.7543 \cdot 10^{-1}$	$5.5622 \cdot 10^{-1}$	$8.6935 \cdot 10^{-2}$	$1.9021 \cdot 10^{-3}$
$\mathbf{SE}(3)$	0.0005	$1.1390 \cdot 10^{-3}$	$1.0167 \cdot 10^{-3}$	$1.5861 \cdot 10^{-7}$	$3.0536 \cdot 10^{-10}$
	0.015	$3.6641 \cdot 10^{-2}$	$3.2604 \cdot 10^{-2}$	$1.6346 \cdot 10^{-4}$	$6.3008 \cdot 10^{-9}$
	0.5	1.1857	1.0576	$1.6580 \cdot 10^{-1}$	$3.1948 \cdot 10^{-3}$
$\mathbf{Sim}(3)$	0.0005	$1.5068 \cdot 10^{-3}$	$1.7052 \cdot 10^{-3}$	$2.9849 \cdot 10^{-7}$	$4.7997 \cdot 10^{-10}$
	0.015	$4.3008 \cdot 10^{-2}$	$4.9284 \cdot 10^{-2}$	$2.4271 \cdot 10^{-4}$	$1.1705 \cdot 10^{-8}$
	0.5	1.5567	1.7407	$2.9783 \cdot 10^{-1}$	$6.8600 \cdot 10^{-3}$

Table 5.1: Accuracy of first, second and third-order approximation of the pose graph Jacobian. At a magnitude of 10^{-10}, the precision of the numerical Jacobians $\mathtt{J}_{\text{num}}$ is reached.

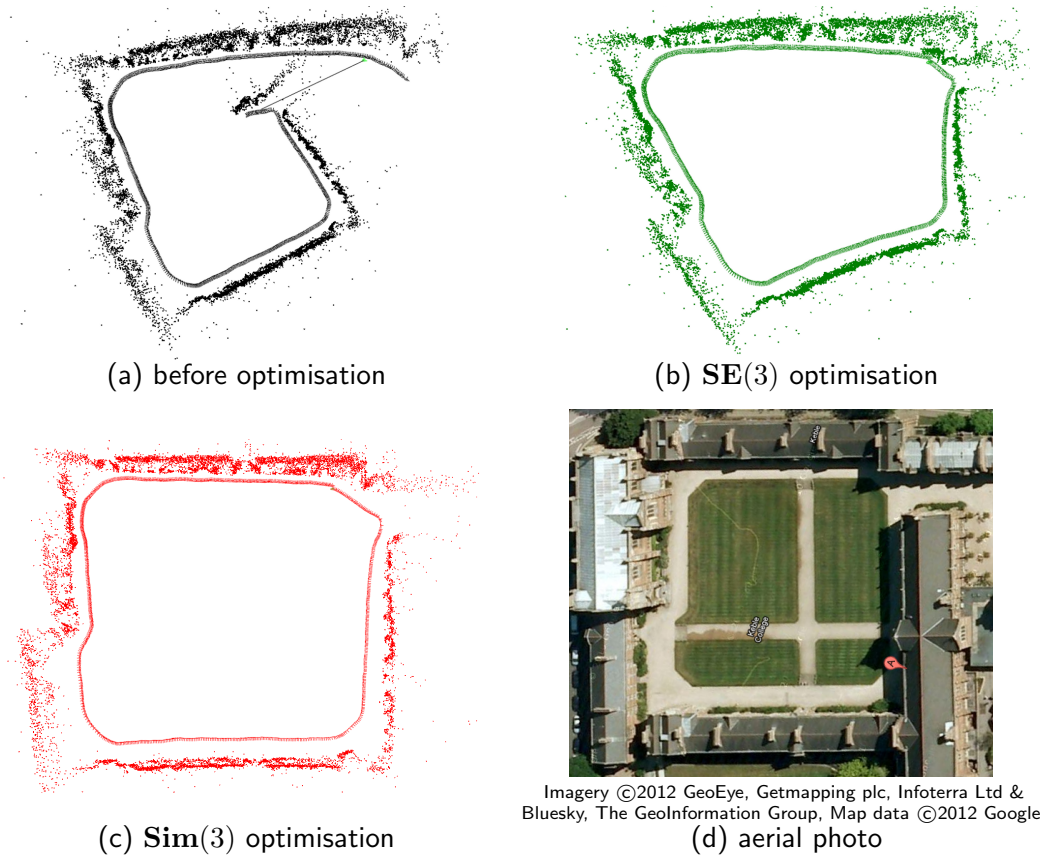
approximate the Jacobian $\mathtt{J}_{\mathtt{A}}^{\mathtt{B}}$ using the kth order Campbell-Baker-Hausdorff expansion. For instance, the third order approximation is:

$$\mathtt{J}_{\mathtt{A}}^{\mathtt{B}} = \left.\frac{\partial \text{cbh}(\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}, \mathbf{d})}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} \approx \mathtt{Ad}_{\mathtt{A}} + \frac{1}{2} \cdot \left.\frac{\partial [\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{0}} \mathtt{Ad}_{\mathtt{A}} + \frac{1}{12} \left(\left.\frac{\partial [\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{0}} \right)^2 \mathtt{Ad}_{\mathtt{A}} \ . \tag{5.22}$$

Its derivation is given in Appendix A.6. This approximation depends on the derivative of the Lie bracket. Lie brackets and their derivatives for $\mathbf{SO}(3)$, $\mathbf{SE}(3)$ and $\mathbf{Sim}(3)$ are presented in Appendix A.3. Note, that under ideal least-square conditions where $\mathbf{d}_{i,j}$ approximates zero, $\mathtt{T}_{ji}\mathtt{T}_i\mathtt{T}_j^{-1} = \mathtt{AB}$ approximates the identity, then $\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{A}\exp(\boldsymbol{\epsilon})\mathtt{B})^{\vee}\big|_{\boldsymbol{\epsilon}=\mathbf{0}}$ approximates the first-order expansion $\frac{\partial}{\partial \boldsymbol{\epsilon}} \text{cbh}(\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}, \mathbf{0})\big|_{\boldsymbol{\epsilon}=\mathbf{0}} = \mathtt{Ad}_{\mathtt{A}}$.

5.4 Experiments

In a first experiment, we analyse the accuracy of the CBH approximations for the pose graph Jacobian $\mathtt{J}_{\mathtt{A}}^{\mathtt{B}}$. We define $\mathtt{A} = \exp(\mathbf{a})$ and $\mathtt{B} = \exp(\mathbf{b})$ with $b_i = -a_i + \mathcal{N}(0, \sigma)$ and $\mathcal{N}(0, \sigma)$ being zero mean normal distributed noise with variance σ^2. In Monte Carlo experiments with one hundred samples each, we compare the numerical Jacobian $\mathtt{J}_{\text{num}}$ (using finite differences) with first, second and third order CBH approximations $\mathtt{J}_1, \mathtt{J}_2, \mathtt{J}_3$. The results for $\mathbf{SO}(3)$, $\mathbf{SE}(3)$ and $\mathbf{Sim}(3)$ are listed in Table 5.1. Under small and medium noise ($\sigma = 0.0005$ and $\sigma = 0.015$), the first order approximation $\mathtt{J}_1$ seems to provide a decent approximation, while the third



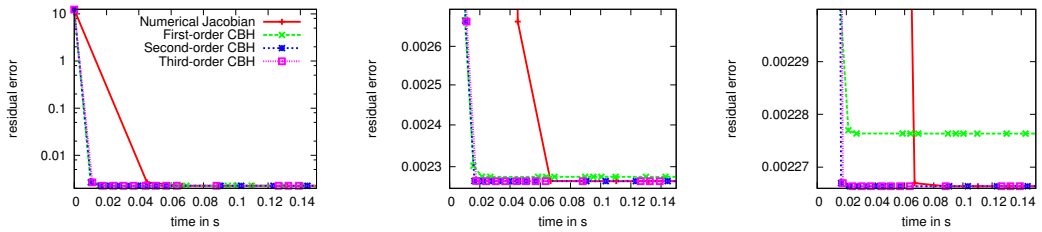
(a) before optimisation (b) **SE**(3) optimisation

(c) **Sim**(3) optimisation (d) aerial photo

Figure 5.3: Keble college data set

order approximation approaches the accuracy of the numerical Jacobian J_{num}. For high noise $\sigma = 0.5$, J_3 still provides a decent approximation.

We evaluate our loop closing framework on the Keble College data set. This time, the monocular exploration framework introduced in Chapter 3 was used to perform an incremental motion estimate of the whole loop as shown in Figure 5.3 (a). It consists of 766 keyframes, 11885 points and 84820 observations. A significant amount of rotational and scale drift is visible. Using the method described in Section 5.3, we detected a loop closure constraint S_{loop} — with a a relative scale change of $1 : 0.37$. The large amount of drift can be partially explained by the fact that the visible scene is always very local in the Keble college data set. Due to the sideway motion, all newly triangulated 3D points leave the field of view rapidly. Also, only a rough intrinsic camera calibration was available. Certainly, future improvements in our sliding-window monocular SLAM framework could lead to a reduction of

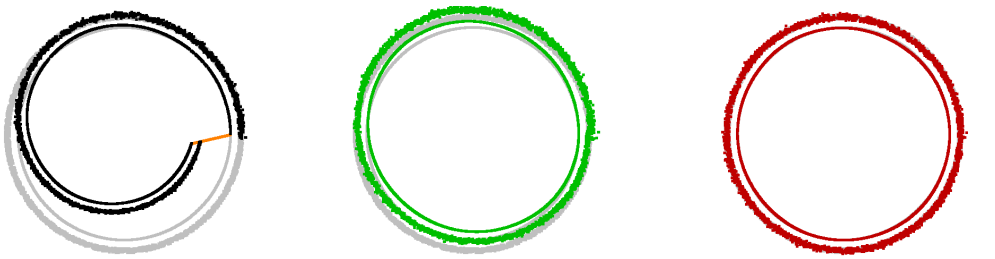


(a) full plot (b) from first iteration (c) from second iteration

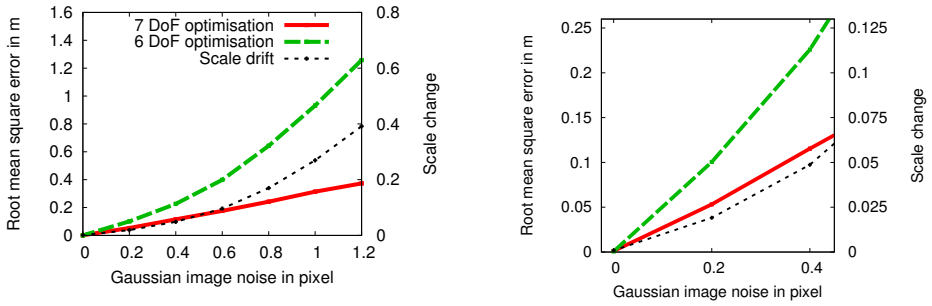
Figure 5.4: Error-versus-cost plots of the similarity graph optimisation using different Jacobian approximation. (a) shows the full plot. (b) and (c) shows close-ups after one and two iterations respectively. One can see that the second and third order CBH-based Jacobians lead to very similar residual errors and computational costs. They converge to the same minima as when employing the numerical Jacobian, but four to five times faster. Using the first-order CBH expansion, however, a near-optimal solution is reached.

drift during exploration. Nevertheless, a certain amount of drift during exploration is unavoidable and our main focus is how to deal with drift when it occurs. A traditional 6 DoF pose-graph optimisation closes the loop but leaves the scale drift unchanged which leads to a deformed trajectory as shown in Figure 5.3 (b). However, if we perform graph optimisation using the similarity transform, the result looks significantly better (see Figure 5.3 (c)). The pose-graph optimisation was performed using $\mathbf{g^2o}$ and a sparse Cholesky solver. In Figure 5.4, the computational cost and accuracy of this $\mathbf{Sim}(3)$ optimisation is shown for various Jacobian approximations. A single iteration brings the energy close to the minimum, taking only 10ms using the CBH-based Jacobians and 45ms using the numerical one. After the pose-graph optimisation, 10 iterations of structure-only BA were performed to refine the points which took 225ms.

In addition to this real-world experiment, we also performed a series of simulation experiments in full 3D space. A simulated camera was moved in a circular trajectory with radius 10m. The camera is directed outwards. A set of 5000 points was drawn from a ring shaped distribution with radius 11m. The camera trajectory consists of 720 poses. In this simulation environment, our monocular exploration framework was applied including feature initialisation and sliding-window bundle adjustment with size 10. Only the camera is not simulated; visual observations are synthetic and data association is given. The difference between the true trajectory and the estimated one is shown in Figure 5.5 (a). In this particular example, we simulated



(a) before optimisation (b) **SE**(3) optimisation (c) **Sim**(3) optimisation

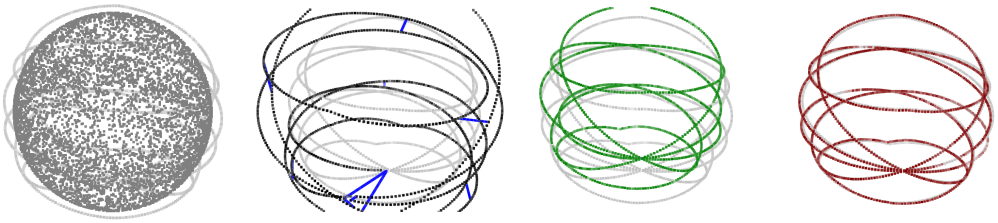


(d) whole plot (e) close-up

Figure 5.5: Monte Carlo experiments on synthetic data

Gaussian image noise with a standard deviation of one pixel. Figure 5.5 (b) and (c) show loop closure correction using **SE**(3) and **Sim**(3). In a Monte Carlo experiment, we varied the amount of image noise from 0 to 1.2 pixels. An important result of our experiment is that there is a clear relation between scale drift and image noise (see Figure 5.5 (d), thin curve). This indicates the correctness of our characterisation of scale as a parameter in SLAM which drifts during exploration in a similar way to rotation and translation. If we define the first pose as our origin, there is still a scale ambiguity of possible maps. Therefore we define a measure between the corrected poses $\mathsf{T}_i^{\text{cor}}$ and the true poses $\mathsf{T}_i^{\text{true}}$ using the minimum of the sum of square differences over the scale s: $M = \min_s \sum_i (\mathbf{t}_i^{\text{true}} - s\mathbf{t}_i^{\text{cor}})^2$. By dividing M by the number of frames and taking the square root, we obtain the root mean square error $RMSE = \sqrt{\frac{M}{720}}$. The average RMSE over the ten simulation runs is shown in Fig. 5.5 (d). One can see that **Sim**(3) optimisation (red curve) outperforms **SE**(3) optimisation (green curve) by a large amount, particularly if the scale change is large. But interestingly, even for small scale changes of one to four percent the **Sim**(3) optimisation performs significantly better than **SE**(3) optimisation (see Fig. 5.5 (e)).

Finally we did an experiment to illustrate that our optimisation framework natu-



(a) aircraft over sphere | (b) before optimisation | (b) **SE**(3) optimisation | (c) **Sim**(3) optimisation

Figure 5.6: Multi-loop-closure example: An aircraft is flying over a sphere and performing SLAM using a single downward-directed camera. The ground truth trajectory is shown in grey.

rally extends to multiple loop closures. Imagine an aircraft flying over a sphere and performing monocular SLAM using a downward directed camera (see Figure 5.6 (a)). Note that no additional prior information regarding the motion and the scene is used: As before, we perform a full 3D SLAM without exploiting the fact that the camera is flying at a constant height over a perfect sphere. After performing monocular exploration, we compute a set of ten loop closure constraints (shown as blue line segments in Figure 5.6 (b)). In this particular example, the **Sim**(3) optimisation leads to a small RMSE of only 0.328, whereas **SE**(3) optimisation results in an RMSE of 2.187 (see Figure 5.6 (c-d)).

5.5 Summary

In this chapter, we have presented a framework for loop closure detection and correction in the context of monocular SLAM. Our approach explicitly acknowledges the issue of scale drift at all stages, and offers a practical way to resolve this drift effectively upon loop closures. The extensive experiments, performed in simulation and using a real outdoor sequence, indicates that a certain amount of scale drift is unavoidable during exploration and this must be taken into account during loop closure to achieve the best results. Furthermore, we have shown how the Jacobians of general pose-graph problems can be efficiently approximated using the nth-order CBH-expansion. Also, we have demonstrated that our optimisation approach naturally extends to multiple loop closures.

5.6 Bibliographic Remarks

Several approaches were presented in the past to apply SLAM in a large-scale scenario. In the context of filter-based SLAM, submapping approaches are predominant. By dividing the map into several submaps, one can not only reduce the overall computational cost, but also reduce the filter uncertainty and therefore the negative influence of non-linearities of the EKF and variants. Early approaches were presented by Tardós et al. (2002), Bosse et al. (2003) and others. Paz et al. (2007) employed EKF submapping using a divide and conquer strategy. This approach lead to a computational complexity which is only linear in the number of landmarks. Clemente et al. (2007) used a submapping technique to large-scale monocular SLAM. Independent submaps are created; by keeping the individual map size limited, the computational cost is bounded during exploration. Loop closures are detected using a scale-invariant map matching approach which aims to detect a subset of features which are geometrically consistent. Relative constraints between the submaps are enforced using an iterated EKF as described by Estrada et al. (2005). A similar monocular SLAM approach is the one of Eade & Drummond (2007). Small submaps, called 'nodes', are estimated using information filters. A graph of similarity constraints is maintained which is optimised using least-squares optimisation.

The approaches above assume that the submaps are statistically *independent* so that no information can be shared between them. Pinies & Tardós (2008) presented a frame-work for *conditionally independent* submapping. Two submaps are only conditionally independent given a set of features present in both submaps, so that updates can be propagated between them in a probabilistically sound way. Thus, no approximation with respect to standard EKF-SLAM is performed.

Another way to adapt filtering to large scale SLAM is to exploit the sparsity in the information matrix. Thrun et al. (2002) showed that while the covariance of the EKF is naturally dense, the information matrix of the extended information filter is approximatively sparse. Their *sparse extended information filter* ignores near-zero blocks stemming from topological distant observations, which leads a more efficient SLAM approach, though with reduced accuracy compared to standard EKF. In a similar approach by Paskin (2003), the information matrix is represented using a junction tree while approximation is achieved by limiting the maximal cluster

size. Walter et al. (2007) introduced the *exactly sparse extended information filter*. By ignoring selected measurements and careful marginalisation of robot poses, the information matrix stays exactly sparse, so that no approximation-induced inconsistency is introduced (apart from the inherent non-linearity issues of filtering). With *smoothing and mapping*, Dellaert & Kaess (2006) went one step further. As in bundle adjustment, neither poses nor points are marginalised so that the related Jacobian stays sparse. Thus, least-squares optimisation can be performed efficiently and inconsistencies are avoided by repetitive relinearisation. Incremental variants of smoothing and mapping were developed by Kaess et al. (2008, 2012). The earlier approach relied on QR factorisation. During exploration, an update only requires a constant number of Givens rotations. At loop closures, however, variable reordering and relinerarisation has to be performed in a batch approach. Kaess et al. (2012) presented the 'Bayes tree', a new graphical model representing the sparse square root information matrix, and employed it for incremental smoothing and mapping. In this formulation, variable reordering and relinearisation can be performed in a full incremental manner.

The concept of pose-graph optimisation originates from Lu & Milios (1997). Gutmann & Konolige (1999) extended their approach by including loop closure detection based on map correlation and an improved incremental estimation scheme. Many other pose-graph methods followed. Eustice et al. (2005) performed pose-graph estimation using exactly sparse information filtering, and applied it in the context of visual underwater SLAM. Olson et al.'s framework (2006) is based on stochastic gradient descent and *relative pose* representations. Their approach dealt well with poorly initialised pose graphs in contrast to previous methods. Grisetti et al. (2007) extended this framework by organising poses in a tree structure so that the computational cost does not increase over time, but only with the map size. Unfortunately, the innovative work of Agrawal (2006) was mainly overlooked by the community. It is an early approach which generalized pose-graph optimization to 3D SLAM — by representing rigid motion using the Lie group $\mathbf{SE}(3)$. Independently, Strasdat et al. (2010b) and Konolige et al. (2010) discussed the close relation between bundle adjustment and pose-graph optimisation, and proposed to solve pose-graph optimisation using Levenberg-Marquardt and sparse Cholesky. The latter demonstrated in a series of experiments the low computational cost as well as the better convergence properties of such a sparse least-squares optimisation approach compared to frameworks using stochastic gradient descent (Grisetti et al., 2007), decomposed

non-linear systems (Frese, 2006) or information filtering (Eustice et al., 2005).

Steder et al. (2007) combined visual odometry with pose-graph optimisation in order to perform globally consistent visual mapping. They combined stereo vision with inertial measurements from an IMU. Absolute roll and pitch measurements were integrated directly, so that poses could be represented using four-dimensional states. In a second setup, they performed visual SLAM on a blimp using a single downward-looking camera. Due to careful control, roll and pitch were kept approximately zero; the problem of scale drift was ignored. Konolige & Agrawal (2008) applied pose graph optimisation to full 3D SLAM using stereo vision. They computed incremental pose updates using 3-point RanSaC, and sliding window bundle adjustment. Afterwards, observations were marginalised out between carefully selected keyframe in order to build a graph of relative poses consttraints. More recent frameworks for large-scale visual SLAM such as Sibley et al. (2009); Mei et al. (2010a) and Lim et al. (2011) are discussed in the following chapter.

This chapter is partially based on Strasdat et al. (2010b)

Chapter 6

Double Window Optimisation

In which we present a general optimisation framework for constant-time visual SLAM, which scales for both local, highly accurate reconstruction and large-scale motion with long loop closures.

Visual SLAM algorithms are approaching performance levels in terms of accuracy, robustness and computational efficiency which now seem close to what would be required for widespread real world applications (Mei et al., 2010a; Lim et al., 2011). Let us recapitulate that several successful early systems (Davison, 2003; Jung & Lacroix, 2003; Eade & Drummond, 2007) solved this inference problem via purely sequential filtering approaches, while the best modern systems work via interleaved tracking and mapping via 'bundle adjustment' optimisation, as pioneered by 'visual odometry' systems such as by Nistér et al. (2006) and Konolige et al. (2007), or Klein & Murray's Parallel Tracking and Mapping (PTAM). The reason for the advantage of this approach, as discussed in Chapter 4, is that the large number of image correspondences which are essential to tracking and mapping accuracy are much more efficiently handled by repeated bundle adjustment optimisation over a selected set of keyframes than by sequential filtering of an uncertain state.

Nevertheless, there remains a divide between visual SLAM systems, not in terms of the fundamental estimation algorithm used but more in the choice of operating domain and the approaches to management of the localisation and mapping process that this implies. On one hand there are systems which target large scale

exploration, usually of outdoor scenes. Originating from work on open-loop visual odometry, these systems now also have the ability to recognise when places are revisited, and handle these loop closures via large scale graph correction (as discussed in the previous chapter).

The other main category is systems designed for *real-time* and very accurate, always metric local mapping, suitable not for exploration but for precise, drift-free localisation in a small domain. This would be what is needed in applications such as augmented reality tracking, or local indoor robot guidance. Here it is assumed that the camera(s) browses a space in a highly repetitive way, and it is necessary to enforce small and medium-sized closures very frequently to maintain the overall consistency of the map. This is achieved by continual map correction in a single metric frame covering the workspace. The classic system here is PTAM (which has, if at all, only been bettered very recently by cutting edge *dense* approaches such as Newcombe et al. (2011b,a)). It runs repeated global bundle adjustment over a spatially selected set of keyframes.

Thus, different motion patterns are addressed using different optimisation approaches. In this chapter, we present a novel and *unified* optimisation framework for visual SLAM which is highly accurate, but at the same time very efficient; it can deal with large scale exploration, long loop closures as well as local browsing motion. We take a *double window* approach that combines accurate pose-point constraints in the primary region of interest with a stabilising periphery of pose-pose soft constraints. Our algorithm automatically builds a suitable connected graph of keyposes and constraints, dynamically selects inner and outer window membership and optimises both simultaneously in a *constant-time* approach. In particular, we borrow the idea of a manifold (Howard et al., 2004; Sibley et al., 2009), representing a neighbourhood metrically and accurately, while relying on the topology of relative relations elsewhere. Furthermore, we present a novel solution for *local registration* by combining *metric* loop closures with top-down feature search in local neighbourhoods of the graph topology. This enables a unified treatment of drift free local browsing and place revisiting after long loops. The framework is applicable for and is tested on various different types of cameras including monocular, stereo and structured light devices.

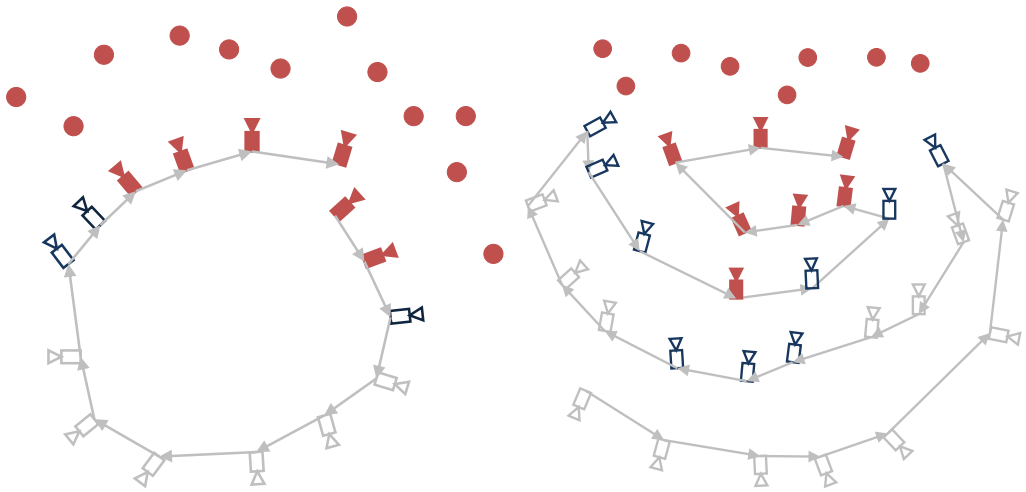
6.1 Optimisation for Visual SLAM

Full bundle adjustment in visual SLAM, while improving rapidly in absolute computation time (Jeong et al., 2010; Konolige, 2010), still suffers from linear to cubic time in the number of variables (depending on particulars of the system), thus limiting its use in large-scale operation. For example, PTAM (Klein & Murray, 2007) runs full bundle adjustment in a background thread, which limits its scale to small workspaces. Our design goal is to have the same accuracy as PTAM in small workspaces while also scaling much better than full BA in handling rapid exploration. There are three main techniques that have been used in the past to tackle this issue:

- Active windows
- Pose-pose reduction
- Relative representations

Active Windows In order to achieve constant-time operation in visual SLAM system, it is common practise to dynamically define a sub-set of all keyframes as the 'active window' over which to apply optimisation. Every keyframe is therefore denoted as either active or inactive at any point in time. There are different strategies possible for the choice of window definition, depending on the camera motion and the system's goals. In visual odometry window frameworks designed for constant time operation during exploration, the active window often consists simply of the most recently captured frames (see Section 3.3.3).

An active window for optimisation must define more than just a set of frames; we must also decide which points to include. One natural selection would be all the points which are visible from the keyframes in the active window. However, this approach has problems. Let us consider one of those points $\mathbf{y}$ at the boundary of the active window which is visible from only two active keyframes, but also from a large number of other keyframes (e.g. eight) outside of the window. The ten observations held for this point would give the potential to triangulate it very accurately in a full joint optimisation. If we optimise its location using only the active window its accuracy will degrade since it is only weakly constrained by the two active keyframes.



(a) Exploration & loop closure (b) Loopy browsing

Figure 6.1: **A**ctive windows. Keyframes within the active window are red (filled); keyframes at the boundary are blue (dark, unfilled); inactive keyframes are grey (light, unfilled). In (a), the camera performs exploration around a loop. The active window has two open ends. In (b), there is very loopy browsing motion. Here, the boundary of the active window consists of many keyframes.

Weakly defined points on the boundary, might lead to weakly defined keyframes and therefore a degeneracy in accuracy.

A common work-around is to *fix* keyframes at the boundary during optimisation (Mouragnon et al., 2006; Lim et al., 2011). First, we include all points $\mathcal{Y}$ which are visible from the active keyframes. Then, we add all other keyframes from which the points $\mathcal{Y}$ are visible as *fixed* keyframes These are used in order to calculate reprojection errors as the points are optimised, but their own poses remain fixed. Fixing keyframes is a common heuristic and works well in exploratory situations with large but few loops as illustrated in Figure 6.1(a).[1] In general, however, it introduces strong bias: Points triangulated from fixed keyframes are very much bound to their positions, and therefore induces strong constraints on the active keyframes nearby. Hence, for a loopy camera motion (Figure 6.1(b)), the number of keyframes at the boundary is relatively large with respect to the total number of keyframes within the active window, and fixing them hampers convergence.

[1] Using an absolute pose parametrisation, this active window should only be fixed at one end. If one uses a relative pose parametrisation (see below), one can fix the window at both ends without hampering convergence too much.

Pose-graph Reduction We have seen in the previous chapter that the procedure of bundle adjusting all frames and points can be approximated using a graph of binary pose-pose constraints. Pose graphs do not reduce the computational complexity of the problem, since (again depending on the particulars of the graph) they have linear to cubic complexity. However, their computational cost is many times smaller than that of BA; and, in practise, their convergence rate is superior too so that pose graph optimisation offers an effective means to close large loops.

A pose graph is an approximation, because binary links between frames do not fully encode the non-linear connections between frames and points. Let us inspect this approximation more closely. The replacement of point-pose constraints by pose-pose constraints is typically understood as marginalisation. Strictly speaking, if we marginalise out a set of points which are visible in m frames, this would lead to a joint Gaussian distribution over all those poses. Instead of representing pose variables with such an *absolute* distribution (i.e. putting an absolute and joint prior on these poses), the joint Gaussian is turned into an m-ary constraint to specify *relative* pose configuration. The introduction of relative pose measurements is the key property of pose graphs; they allow us to re-linearise and therefore perform inference in terms of non-linear optimisation. In all predominant approaches, m-ary constraints are approximated by a set of binary constraints which link all poses pairwise. This approximation proved to be very effective since it significantly reduces the computational complexity, and it leads to very accurate results. Its theoretical justification has not yet been fully understood; however, it seem to be linked to the procedure of turning absolute distributions into relative constraints.

In practise, the question remains of when to join two poses by a binary constraint. In the previous chapter, we followed the simple heuristic of Konolige & Agrawal (2008) by adding constraints along the path of motion plus a few loop closure constraints. In this work, we employ the concept of *covisibilty* (Mei et al., 2010b); in this more rigorous approach, we connect all those poses which have a significant scene overlap.

Relative Representations Instead of representing poses with respect to a global frame of reference, one can alternatively use a relative parametrisation. Note that we are not talking about relative pose *constraints* which can be seen as measurements in the context of pose graph optimisation, but about a relative formulation of the

pose variables themselves — the entities we are modifying during optimisation.[2]

One prominent example is Sibley et al.'s Relative Bundle Adjustment (RBA, 2009) which uses a relative representation for frame and point variables. Since the global position of the frames is not computed, this must be recovered from the relative variables, and involves significant computation. However, Sibley et al. argue that there is usually no need for full reconstruction, and this argument aligns with our notion of a manifold, in which metric reconstruction occurs only in a local region. To work in constant time, RBA makes the active window assumption. RBA is equivalent to standard bundle adjustment if the network of relative poses form a tree. Thus, it works especially well on exploratory scenarios where there are no cycles *within* the active window (Figure 6.1(a)). However, if there are loops within the active window, the accuracy degrades as it does not enforce the condition that relative pose transformations around the loop add up to the identity.

As we will see shortly, our double window formulation is somewhat between a fully relative formulation such as RBA and the absolute formulation of standard BA. Poses within an active window are represented with respect to a common reference frame so that metric consistency is fulfilled. Poses outside the active window are merely defined by a set of relative pose-pose constraints.

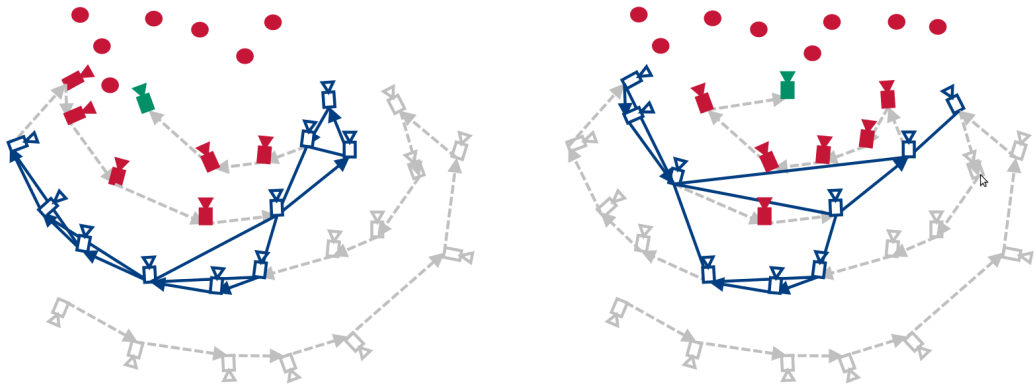
6.2 Double Window Optimisation Framework

In this section, we introduce our scalable back-end for visual SLAM. For an example, we will concentrate on stereo SLAM. In Section 6.2.7, we explain how this framework is extended to monocular SLAM — including appropriate treatment of scale drift in constant time.

6.2.1 Overview

In order to achieve scalable, usually constant-time performance, we apply an active window scheme. The novelty of our framework is the fact that we use a *double window* approach. An inner window of point-pose constraints (as in bundle adjust-

[2] Indeed, there are pose graph optimisation frameworks such as Grisetti et al. (2007) where not only the constraints but also the pose variables are relative.



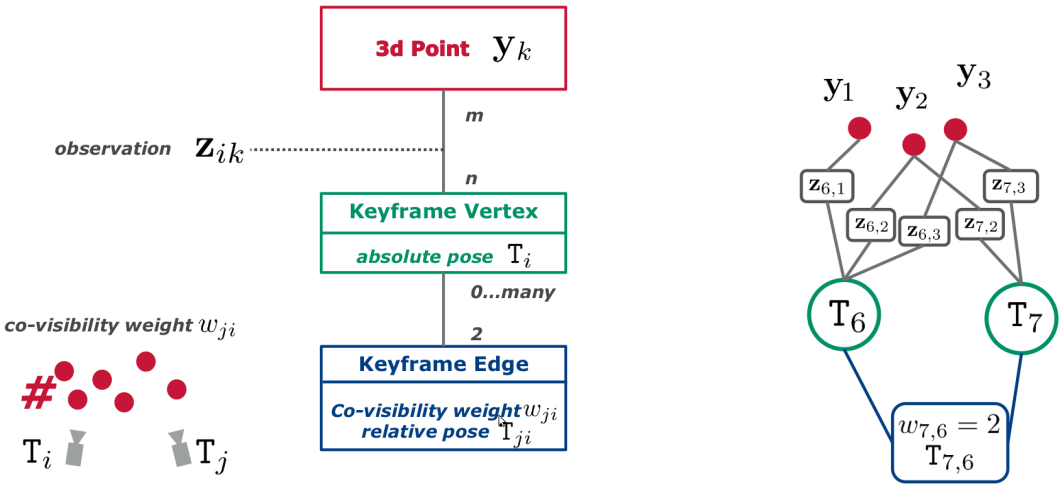
(a) Example graph (time step 1) (b) Example graph (time step 2)

Figure 6.2: Illustrations of the Double Window Optimization (DWO) framework. Keyframes and points in the inner window are shown in red, while keyframes in the outer window are shown in blue. The current reference keyframe is shown in green.

ment) is supported by an outer window of pose-pose constraints (as in pose graph optimisation). Pose-pose constraints are defined by *covisiblity* (Mei et al., 2010b). Two poses are connected to each other if they share enough common features. As opposed to the approach we described in the previous chapter where we used sliding window BA for exploration and then pose-graph optimization for loop closing, we couple the point-pose constraints and the pose-pose constraints within a single optimisation framework. While the inner window serves to model the local area as accurately as possible, the pose-graph in the outer window acts to stabilise the periphery. The soft constraints of the periphery contrast with fixed keyframes within a (relative) BA approach, which are hard constraints. An example graph is illustrated in Figure 6.2.

6.2.2 The SLAM Graph Structure

The SLAM graph consists of a set of keyframe vertices $\mathcal{V}$, a set of 3D points $\mathcal{P}$, and a set of relative edges $\mathcal{E}$. Each keyframe vertex V_i saves its *absolute* pose $\mathtt{T}_i$, remembers which points $\mathbf{y}_k \in \mathcal{P}$ are visible from $\mathtt{T}_i$ and also saves all corresponding observations $\mathbf{z}_{ik}$. An edge E_{ij} between two pose vertices V_i and V_j has a covisibility weight w_{ij}, which is the number of points which are visible both in V_i and V_j. Also, an edge is marked as being marginalised or not. If it is marginalised it also stores



(a) UML structure diagram (b) example graph

Figure 6.3: SLAM graph structure used for the double window framework.

the relative pose constraint $\mathtt{T}_{ij}$ between $\mathtt{T}_i$ and $\mathtt{T}_j$. Otherwise, the relative pose is implicitly defined as $\mathtt{T}_{ij} = \mathtt{T}_i \cdot \mathtt{T}_j^{-1}$. A sample graph is visualized in Figure 6.2(a). At all times, exactly one keyframe vertex is labelled as being the *reference keyframe* V_{ref} . This is usually the keyframe which is added last, but could also be an older keyframe which is revisited during loopy motion.[3]

A UML (Unified Modelling Language) structure diagram and a corresponding example graph are shown in Figure 6.3.

6.2.3 Optimisation and Marginalisation

To construct the double-window structure, we start from the reference keyframe V_{ref}, and perform a weighted breath-first search over the neighbours of V_{ref}, in such a way that the neighbour with the highest covisibility weight w_{ij} is selected first. The first M_1 keyframes are considered as being part of the inner window $\mathcal{W}_1$, whereas the following M_2 keyframes are members of the outer window $\mathcal{W}_2$ (typically $M_1 << M_2$). All points visible from the inner window are included in the optimisation. Thus, all frames in the inner window $\mathcal{W}_1$, and some frames in the outer window

[3]Despite calling V_{ref} the reference keyframe, its absolute pose $\mathtt{T}_{\text{ref}}$ needs not to be the identity. Its name stems from the fact that it is the current centre of interest. The inner and outer window should shape around it as shown in Figure 6.2.

$\mathcal{W}_2$ are connected with point-pose constraints $\mathbf{z}_{ik}$ to the set of points as is usually done in BA. In addition, all frames in the outer window are connected to their local neighbours using pose-pose constraints $\mathtt{T}_{ji}$ as in pose-graph optimisation. This results in the following cost function:

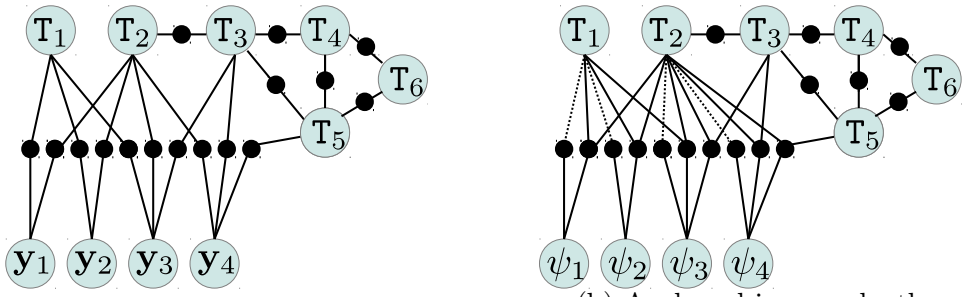
$$\chi^2 = \sum_{\mathbf{z}_{ik}} (\mathbf{z}_{ik} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathbf{y}_k))^2 + \sum_{\mathtt{T}_{j,i}} \boldsymbol{\upsilon}_{ji}^\top \Lambda_{\mathtt{T}_{ji}} \boldsymbol{\upsilon}_{ji} \ . \tag{6.1}$$

Here, $\boldsymbol{\upsilon}_{ji} := \log(\mathtt{T}_{ji} \cdot \mathtt{T}_i \cdot \mathtt{T}_j^{-1})^\vee_{\mathbf{SE}(3)}$ is the relative pose error in the tangent space of $\mathbf{SE}(3)$. A corresponding factor graph is shown in Figure 6.4(a). The matrix $\Lambda_{\mathtt{T}_{ji}}$ in equation (6.1) is the inverse covariance of the binary constraint $\mathtt{T}_{ji}$. Instead of estimating this uncertainty accurately using proper marginalisation (Konolige & Agrawal, 2008), we suggest to approximate $\Lambda_{\mathtt{T}_{ji}}$ coarsely:

$$\Lambda_{\mathtt{T}_{ji}} = w_{ij} \begin{bmatrix} \lambda_{\text{trans}}^2 \mathtt{I}_{3\times3} & \mathtt{0}_{3\times3} \\ \mathtt{0}_{3\times3} & \lambda_{\text{rot}}^2 \mathtt{I}_{3\times3} \end{bmatrix} . \tag{6.2}$$

While the rotational component λ_{rot} is a constant, the translational λ_{trans} component is chosen to be proportional to the parallax of $\mathtt{T}_{ji}$ — the translation $\mathbf{t}_{ji}$ normalised by the average scene depth. As we will see in the experimental section, this efficient approximation of $\Lambda_{\mathtt{T}_{ji}}$ leads to very accurate results. Furthermore, we were not able to reproduce significantly better results using proper pair-wise marginalisation instead. We believe the reason for this is twofold: On one hand, turning BA into a binary pose graph is an approximation per se, because the marginalisation of a landmark visible in N frames should ideally lead to an hyper-edge jointly connecting all those frames. On the other hand, the pose-pose network we use embodies a covisibility graph with typically many inter-connections (such as in Figure 6.9). We believe that the accuracy supported by this structure overwhelms the approximation due to the use of diagonal precision matrices.

Double window optimisation is performed by minimising the sum of squared error χ^2 with respect to all poses $\mathtt{T}_i \in \mathcal{W}_1 \cup \mathcal{W}_2$ in the double window and all corresponding points $\mathbf{y}_k$. First and second order sparsity is taken into account, and the optimisation is performed using $\mathbf{g^2o}$ (Kümmerle et al., 2011a). During optimisation, we do not define a fixed origin, since fixing a keyframe as the global origin can seriously degrade convergence if the selected keyframe is badly localized relative to its neighbours. Instead, we let the damping factor of Levenberg-Marquardt take care of the gauge freedom.



(a) Standard parametrisation

(b) Anchored inverse depth points

Figure 6.4: Factor graphs of double window optimisation. In (a), points are represented with respect to a global coordinate frame. Point observations as well as pose-pose relations result in binary constraints. In (b), we illustrate the improved parametrisation using anchored inverse depth points. In general, observations usually result in ternary constraints; but observations in their anchor frame are unary constraints.

As the camera moves in space, the reference pose $\mathtt{T}_{\text{ref}}$ changes as well as the configuration of the inner and outer window (see Figure 6.2). When a keyframe $\mathtt{T}_j$ is added to the double window — i.e. $\mathtt{T}_i \notin \mathcal{W}_1^{\mathbf{old}} \cup \mathcal{W}_2^{\mathbf{old}}$, but $\mathtt{T}_i \in \mathcal{W}_1^{\mathbf{new}} \cup \mathcal{W}_2^{\mathbf{new}}$ — we need to make sure it is well initialized before we perform the joint optimisation. Starting from the reference pose $\mathtt{T}_{\text{ref}}$, we initialise

$$\mathtt{T}_j = \mathtt{T}_{jk} \cdot \ldots \cdot \mathtt{T}_{ka} \cdot \mathtt{T}_{\text{ref}} \tag{6.3}$$

along the path of relative pose constraints $\mathtt{T}_{jk} \cdot \ldots \cdot \mathtt{T}_{ka}$ which connects $\mathtt{T}_j$ with $\mathtt{T}_{\text{ref}}$. This is done to ensure that the pose $\mathtt{T}_j$ is well localised relative to its neighbours in the inner window.

However, there is a caveat in the case that the relative constraints form a large loop *within* the double window. If the loop constraints do not add up to the identity (due to estimation errors), we need to make sure that the loop does not break within the inner window — in order to abet low reprojection errors and thus high accuracy. To achieve this, we use the following strategy: First, breath-first traversal is performed on $\mathcal{W}_1^{\mathbf{new}} \cup \mathcal{W}_2^{\mathbf{new}}$ to create a spanning tree with root $\mathtt{T}_{\text{ref}}$. This tree connects all members in the new double window, with $\mathtt{T}_{\text{ref}}$ being in the centre. Now, we do not only reinitialise all poses $\mathtt{T}_i \notin \mathcal{W}_1^{\mathbf{old}} \cup \mathcal{W}_2^{\mathbf{old}}$, but also all of their children. In this way, we will make sure that a loop will break in the outer periphery of the outer window. In order to make sure that the points in the inner window are also

localised well, we perform a few iterations of structure-only BA, which can be done very efficiently.

Let us recapitulate that a frame in the inner window is defined by its absolute pose; all other frames are merely defined by relative constraints which connect them to their neighbours. Once a frame leaves the inner window, its relative constraints need to be recomputed since its configuration relative to its neighbours might have changed. Thus, let $\mathtt{T}_i \in \mathcal{W}_1^{\mathbf{old}}$ and $\mathtt{T}_j \in \mathcal{W}_1^{\mathbf{old}}$. If $\mathtt{T}_i \notin \mathcal{W}_1^{\mathbf{new}}$ or $\mathtt{T}_j \notin \mathcal{W}_1^{\mathbf{new}}$ we compute $\mathtt{T}_{ji} = \mathtt{T}_j \cdot \mathtt{T}_i^{-1}$ and $\Lambda_{\mathtt{T}_{ji}}$ as explained above.

6.2.4 Improved Point Parametrisation using Anchored Inverse Depth Features

So far, we considered a point parametrisation in which points are stored using standard Euclidean 3-vectors with respect to a common world reference frame. This approach is appealing through its simplicity, since no special point management is required. However, it has two major drawbacks: First, the projection of Euclidean points is highly non-linear, and this slows down the joint structure and motion estimation in bundle adjustment significantly.[4] Second, while structure-only bundle adjustment is usually very effective to triangulate a point with respect to a set of frames, this optimisation, as well as joint bundle adjustment, is doomed to diverge catastrophically if the initial point estimate lies behind any of those frames. This, however, can easily happen in a large scale scenario since the absolute poses of keyframes might change drastically once they re-enter the double window as we discussed previously. Therefore, we suggest to use an improved parametrisation where points are stored as anchored inverse depth features (as in Sibley et al., 2009; Lim et al., 2011). Each point is represented using inverse depth parameters $\boldsymbol{\psi}_k$ and anchored to the frame $\mathtt{T}_{a(k)}$ in which it was seen first. We modify equation (6.1) and end up with the following energy:

$$\chi^2 = \sum_{\mathbf{z}_{ik}} (\mathbf{z}_{ik} - \hat{\mathbf{z}}(\mathtt{T}_i \cdot \mathtt{T}_{a(k)}^{-1} \cdot \Pi(\boldsymbol{\psi}_k)))^2 + \sum_{\mathtt{T}_{j,i}} \boldsymbol{\upsilon}_{ji}^\top \Lambda_{\mathtt{T}_{ji}} \boldsymbol{\upsilon}_{ji} \ . \tag{6.4}$$

[4] There are two effective workarounds. First, one could represent the points using homogeneous coordinates, thus as members of the 3-sphere. This more linear parametrisation is well known for leading to faster convergence (Triggs et al., 1999). The incremental update can be specified in the tangent space of the 3-sphere (Hartley & Zisserman, 2004, pp.624). Or one could speed-up the point convergence within bundle adjustment using *embedded point iteration* (Jeong et al., 2010).

The corresponding factor graph is illustrated in Figure 6.4(b). As one can see, anchored point observations result in ternary constraints. If, however $\mathtt{T}_i = \mathtt{T}_{a(k)}$, the prediction simplifies to $\hat{\mathbf{z}}(\Pi(\boldsymbol{\psi}_k))$ so that the observation of a point in its anchor frame can be represented more concisely using a unary constraint.

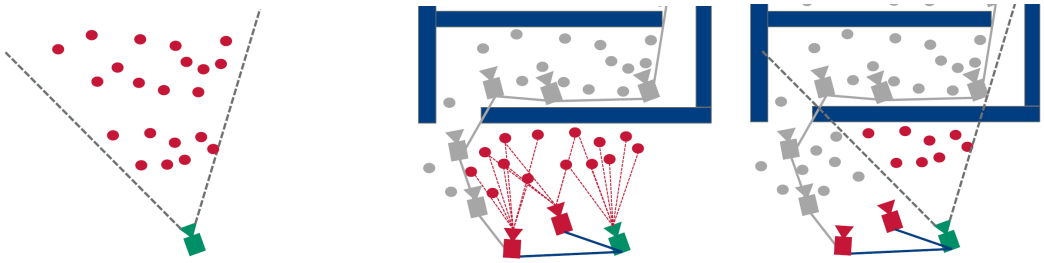
This anchored inverse depth parametrisation is beneficial since its points are always well localized relative to their anchor frame; in practice, the point-behind-frame problem does not occur. Furthermore, the inverse depth parametrisation leads to faster convergence. However, there is one complication: A point $\boldsymbol{\psi}_k$ can only be used in the double window optimisation if the corresponding anchor frame $\mathtt{T}_{a(k)}$ is included. We use the following strategy: For each point $\boldsymbol{\psi}_k$ which is visible from the inner window, we identify its anchor frame $\mathtt{T}_{a(k)}$. If $\mathtt{T}_{a(k)}$ is in the inner window, we can safely include $\boldsymbol{\psi}_k$ in the optimisation. Otherwise, we check whether there is a direct edge $E_{a(k),b}$ between the anchor frame $\mathtt{T}_{a(k)}$ and a frame $\mathtt{T}_b \in \mathcal{W}_1$ in the inner window. This ensures that the anchor is still accurately defined. If this is the case, we can include the point $\boldsymbol{\psi}_k$; its anchor frame $\mathtt{T}_{a(k)}$ is added to the outer window.

6.2.5 Candidate Points Set for Top-down Tracking

For ego motion estimation, we seek to detect a set of 3D points in the current image. In PTAM, all 3D points in the map are potential candidates for tracking. This strategy is suitable for small workspaces, but does not scale very well with for large scale mapping. In previous large scale SLAM frameworks (e.g., Mouragnon et al., 2006; Konolige & Agrawal, 2008; Strasdat et al., 2010b; Lim et al., 2011), features are either tracked using a purely bottom-up visual odometry approach or points from the last m frame or keyframes are considered. Instead, we select points which are visible from the topological neighbourhood around the reference keyframe V_{ref}. This local neighbourhood $\mathcal{N}_1$ consists of all keyframes V_i connected to V_{ref} including itself:

$$\mathcal{N}_1 := \{V_i : E_{\text{ref},i} \in \mathcal{E}\} \cup \{V_{\text{ref}}\}. \tag{6.5}$$

All points visible from these frames are considered as potential candidates for tracking. As in PTAM, a point is only actively searched for if its reprojection lies within the current image boundaries, it is not too far or too close, and is not seen from a too different viewing angle compared to its initial observation.



(a) Project all points (b) Project points from local neighbourhood

Figure 6.5: Construction of point candidate set for pose tracking. In the naive approach (a) all points are selected which project into the reference frame. In our scalable approach (b), we only consider points visible from the local neighbourhood. Points which are occluded are unlikely to be considered.

Apart from scalability, selecting points using the local neighbourhood of frames has another advantage over PTAM's approach: it implicitly takes care of occlusion (see Figure 6.5). Points which are occluded in the current frame are probably not visible from nearby frames either.

6.2.6 Adding New Keyframes and Local Registration

If a certain criterion is fulfilled (e.g. the incremental translation/parallax exceeded a threshold, or the number of tracked feature dropped below a critical limit) the current video frame is added as a new keyframe V_i to the graph. For all keyframes V_j in the graph which share at least Θ (typically Θ being 15 to 30) covisible points with the current frame, we include an edge E_{ji}, mark it as unmarginialized and assign a corresponding covisibility weight w_{ji}. This *local registration* ensures that the new keyframe V_i is connected to keyframes with significant scene overlap. Further registration is performed using metric loop closure as will be elaborated below. Finally the new keyframe is chosen to be the new reference keyframe $V_{\text{ref}} := V_i$.

6.2.7 Extension to Monocular SLAM

In monocular SLAM there is a scale ambiguity. For BA, this has no particular consequences apart from the fact that the overall gauge freedom increases from six DoF to seven DoF. However, in pose graph optimisation more care has to be

taken. As we have seen in the previous chapter, relative **SE**(3) constraints cannot represent scale change in a pose graph. Thus, we use seven DoF constraints to correct for rotational, translational and scale drift.

Since significant scale drift only occurs along large loops, and we are interested in a constant-time treatment of scale drift, we apply the following heuristic. The absolute poses $\mathtt{T}_i$ as well as the relative poses $\mathtt{T}_{ij}$ are members of **Sim**(3) instead of **SE**(3). However, the scale parameter s remains fixed most of the time. When a new keyframe is added to the graph, the corresponding scale is set to $s = 1$. There is only one case when a relative scale $s \neq 1$ is introduced: at large-scale, appearance-based loop closures (see Section 6.4.2). In particular, we use a RanSaC scheme to determine the rotation $\mathtt{R}$, translation $\mathbf{t}$, and scale s of the relative loop closure constraint. This scale change gets propagated once poses are reinitialised using equation (6.3).

6.3 Visual Frontends

In order to evaluate the double window optimisation framework not only on synthetic data, but also using real-image sequences, a visual front-end for the corresponding sensor is required.

Our monocular front-end is based on PTAM since this is probably the most robust and best tested front-end freely available.[5] We adapt the PTAM front-end to the needs of the DWO back-end. 3D candidate points for tracking are estimated as described in Section 6.2.5. Epipolar search for feature initialisation is only performed between keyframes which are connected with an edge. In general, all `for` loops in PTAM which iterate over all points or all keyframes are replaced. Instead, sets of points and frames are accessed along the local connectivity of the SLAM graph.

In the case of stereo SLAM, we use a custom front-end which exploits the advantages of stereo-cameras as well as the computational power of modern GPUs. Given a pair of rectified frames I_l, I_r, we estimate the disparity $d = u_l - u_r$ for each pixel in the left frame. This can be done very efficiently – either on the GPU or on the

[5] The source code of PTAM is available from http://www.robots.ox.ac.uk/~gk/PTAM/ under a custom license for non-commercial use.

CPU – using the block matching stereo algorithm implemented in OpenCV.[6] This efficient method works if the scene is textured sufficiently. For settings with many untextured regions, stereo method using variational optimisation should be used instead (e.g. Ranftl et al., 2012). Once the disparity is estimated for a pixel (u_l, v_l) in the left reference frame, we can calculate the corresponding 3D point $\mathbf{y}_{u_l,v_l}$:

$$\mathbf{y}_{u_l,v_l} = \begin{pmatrix} \frac{y_3}{f}(u_l - p_1) \\ \frac{y_3}{f}(v_l - p_2) \\ y_3 \end{pmatrix} \quad \text{with} \quad y_3 = \frac{b}{f \cdot d} \,. \tag{6.6}$$

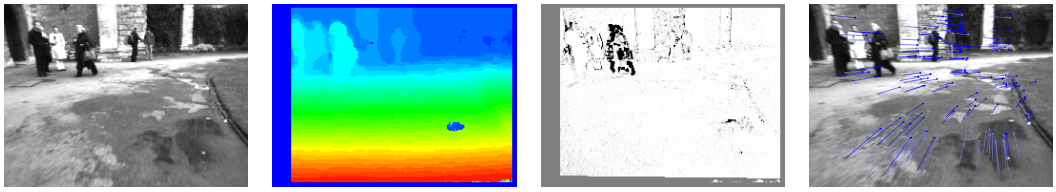
with f being the focal length, $\mathbf{p}$ being the principal point and b being the baseline of the calibrated stereo rig. Thus, we create a *dense* model of 3D points with respect to the camera pose $\mathtt{T}_n$. Now, we wish to estimate the camera pose $\mathtt{T}_{n+1}$ of the subsequent (left) frame $I_l^{[n+1]}$ in order to guide the feature tracking. Given the dense point model, we perform Lucas-Kanade tracking (Baker & Matthews, 2004). In particular, we follow the approach of Newcombe, Lovegrove and Davison (2011b) and minimise the following photometric energy:

$$\chi^2 = \sum_{u,v} \rho \left(\left(I_l^{[n]}(u,v) - I_l^{[n+1]}(\hat{\mathbf{z}}_{\text{mono}}(\mathtt{T}_{n+1} \cdot \mathbf{y}_{u,v})) \right)^2 \right) \tag{6.7}$$

with respect to the camera pose $\mathtt{T}_{n+1} \in \mathbf{SE}(3)$; here, ρ is a robust kernel. One way to interpret this least-squares method is the following: It estimates an optical flow field which is forced to be consistent with the dense point model as well as with a rigid body motion. The kernel ρ is used in order to be robust to pixels where this assumption is violated. This can be due to violation of the static scene assumption, wrong disparity estimates, specular highlights etc. The least-squares optimisation can be performed very efficiently on a modern GPU. A pyramidal approach is used in order to achieve a large basin of convergence, so that this incremental pose estimator can deal with very rapid motion. Since this leads to an accurate pose estimate already, so guided feature tracking can be performed within small circular regions (i.e. 1-3 pixel radius). Afterwards, the pose is refined using motion-only bundle adjustment on the sparse point cloud. This way we make sure that the pose remains consistent to the 3D points, which are estimated using double window optimisation. This hybrid dense/sparse tracking is illustrated in Figure 6.6.

In each keyframe, we evaluate using a quadtree (see Section 3.4.3) where new 3D points should be initialised. These candidate points are immediately used for pose

[6] http://code.opencv.org/



(a) left image (b) disparity image (c) residual image (d) feature tracks

Figure 6.6: Given a stereo pair — the left frame is shown in (a) — we calculate a disparity image (b). Using dense Lucas-Kanade tracking, the incremental pose is estimated. The corresponding residual errors are shown in (c). The error is high (black) where the static scene assumption is violated, e.g. moving woman (top left), or the disparity estimate is wrong, e.g. puddle (bottom right). This dense tracking leads to accurate ego motion estimate, which makes the sparse point tracking (d) very efficient and robust.

tracking. However, we only add them to the map if they are redetected in one of the subsequent keyframes. In this way we make sure that only high-quality feature points which could be detected repeatedly are added to the map.

6.4 Loop Closures

Keyframes are locally registered to each other by including new edges in the SLAM graph. A large portion of this registration happens when new keyframes are added (Section 6.2.6). The remaining registration is tackled by loop closure events. We distinguish between two types of loop closures. The first type is local loop closures which can still be detected metrically. The second type is large loop closures which are detected using appearance-based place recognition.

6.4.1 Metric Loop Closure

Checking for metric loop closures is done by searching for 3D points in the reference keyframe V_{ref} which are *not* visible from its neighbourhood $\mathcal{N}_1$ (see equation (6.5)). The process is illustrated in Figure 6.7. First, we determine a larger neighbourhood $\mathcal{N}_2$ around V_{ref} using weighted breath-first search (as described in Section 6.2.3). We construct a set $\mathcal{A}$ of potential loop closure points by selecting

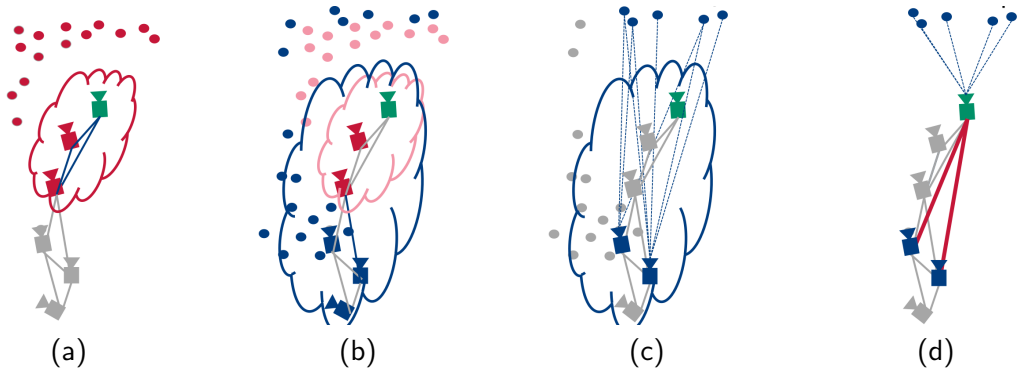


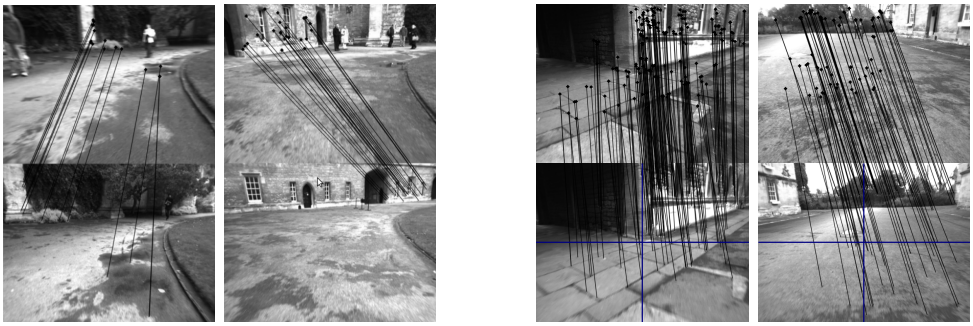
Figure 6.7: Metric loop closure: In (a), we first determine all points (=red dots) visible from the local neighbourhood $\mathcal{N}_1$ of the reference keyframe. Afterwards, in (b), we construct a wider neighbourhood $\mathcal{N}_2$. We determine a set of (blue) points which are visible from $\mathcal{N}_2$ but not $\mathcal{N}_1$. In (c), we determine the subset of those (blue) points which indeed are visible in the reference keyframe. Finally, in (d), we add new edges between the reference keyframe V_{ref} and keyframes V_i in the wider neighbourhood if at least Θ of those points visible in V_{ref} are also visible in V_i.

points which are visible in $\mathcal{N}_2$, but not in $\mathcal{N}_1$. Then we try to measure such a point $\mathbf{y} \in \mathcal{A}$ in the reference keyframe using guided search on a larger search radius (e.g. 10 pixels). If enough of those points are found, we minimise their reprojection error with respect to a common pose T_{loop} using robust motion-only bundle adjustment, starting from T_{ref} as the initial guess. Afterwards, we prune all points from $\mathcal{A}$ whose reprojection error exceeds a threshold (e.g. one pixel). The motivation for this approach is the following: We only want to consider a set of points for metric loop closure if their observations in V_{ref} *mutually agree* with each other, hence can be explained by a common pose T_{loop}.

Then, for each keyframe $V_i \in \mathcal{N}_2 \backslash \mathcal{N}_1$ we check how many points in $\mathcal{A}$ are also visible in V_{ref}. If there are Θ or more co-visible points between V_i and V_{ref}, we have detected a metric loop closure, and we include a new edge $E_{\text{ref},i}$. This edge is marked as being marginalised, and the corresponding pose constraint $\mathsf{T}_{\text{ref},i}$ is set:

$$\mathsf{T}_{\text{ref},i} := \mathsf{T}_{\text{loop}} \cdot \mathsf{T}_i^{-1}. \tag{6.8}$$

Note that if there is some drift along the chain $\mathsf{T}_i \cdot \mathsf{T}_{a_1 a_2} \cdot \ldots \cdot \mathsf{T}_{a_{n-1} a_n} \cdot \mathsf{T}_{\text{ref}}$, then $\mathsf{T}_{\text{loop}} \neq \mathsf{T}_{\text{ref}}$ and the residual $\boldsymbol{\upsilon}_{\text{ref},i} = \log(\mathsf{T}_{\text{ref},i} \mathsf{T}_{\text{ref}} \mathsf{T}_i^{-1})$ might be large, something which will be resolved during the next round of double window optimisation.



(a) large view-point changes (b) (c) more conservative (d)

Figure 6.8: Examples of metric loop closures.

These metric loop closures lead to an effective scheme for local registration. When we follow this approach, almost each keyframe pair which has some scene overlap is connected with an edge. Even for significant view-pointed changes, we can detect loop closures as shown in Figure 6.8(a,b). However, this way, many edges are added to the SLAM graph which are only weakly defined, thus the relative pose is inaccurate. This is especially true if the feature matches are only located in the image background as in Figure 6.8(b)[7]. Being too lavish with metric loop closures increases the SLAM graph connectivity and therefore the computational cost of the optimisation; at the same time, an accuracy benefit is unlikely since many pose constraints are only weakly defined in this case. Therefore we suggest a heuristic which avoids weakly defined constraints by ensuring that loop closure correspondences are well-spread in the reference keyframe. We only accept the metric loop closure if there are at least $\frac{1}{2}\Theta$ matches in the upper, lower, right and left half of the reference image. Typical loop closures using this more conservative approach are shown in Figure 6.8(c,d).

6.4.2 Large-scale Loop Closure

Candidates for large loop closures can be efficiently detected using appearance information only. We use the following approach. First, we perform an offline training procedure. From the INRIA holiday image dataset[8], we learn a dictionary of *visual*

[7] Cadena et al. (2011) discussed the benefits to use near as well as far visual information for loop closure detection.

[8] http://lear.inrialpes.fr/~jegou/data.php

words. In particular, we detect SURF keypoints (Bay et al., 2006) in every image. We rescale the image adaptively such that 500 to 2000 keypoints are detected. For all these keypoints in all training images (e.g. 150) we calculate SURF descriptors. Using the hierarchical K-means clustering approach of Muja & Lowe (2009), we detect approximately 10,000 clusters (=visual words) in the 64-dimensional space of SURF descriptors.

During SLAM, we calculate SURF keypoints and descriptors for each new keyframe. We match those keypoints against the dictionary using an approximate nearest neighbour search (Muja & Lowe, 2009). Then we employ the well-known *term frequency-inverse document frequency* (tf-idf) statistic, popularised by Sivic & Zisserman (2003) in the context of visual recognition, to hypothesise loop closures. Compared to pure appearance-based approaches, we thereby exploit the topology of our SLAM graph. Especially, we never try to detect an appearance-based loop closure between two keyframes which are topologically close. In such a case the metric detection scheme is more effective. Once a loop closure hypothesis is generated between two keyframes V_a and V_b, we verify it using a geometric consistency check based on a three point RanSaC scheme. In the case of monocular SLAM, the depths of the SURF keypoints are approximated using k-nearest neighbour regression (Section 5.3.1). Given a set of three 3D-3D correspondences, a relative pose $\mathtt{T}_{a,b} \in \mathbf{Sim}(3)$ is uniquely defined.

For stereo SLAM, we modify the model of Arun et al. (1987) which we presented in equation (5.11-5.14) by setting the scale to one and thus

$$\mathtt{R} = \mathtt{V}\mathtt{U}^\top, \quad \mathbf{t} = \mathbf{c} - \mathtt{R}\bar{\mathbf{c}} \; . \tag{6.9}$$

If more than Θ inliers are found, the loop closure is accepted and an edge $E_{a,b}$ is added the graph.

6.5 Experiments

6.5.1 Simulation Experiments

The main motivation for the double-window optimisation (DWO) approach is that it can deal with different motion patterns. In particular, it can smoothly handle

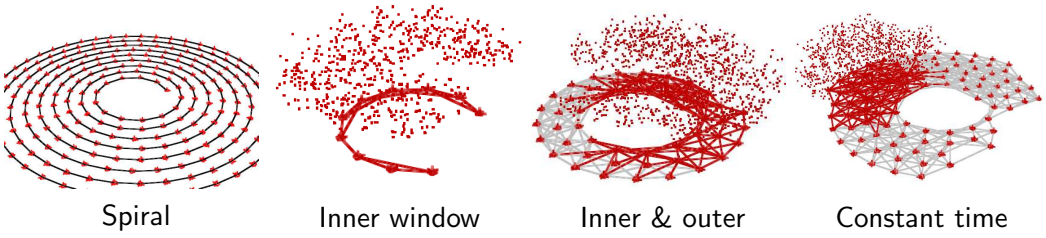
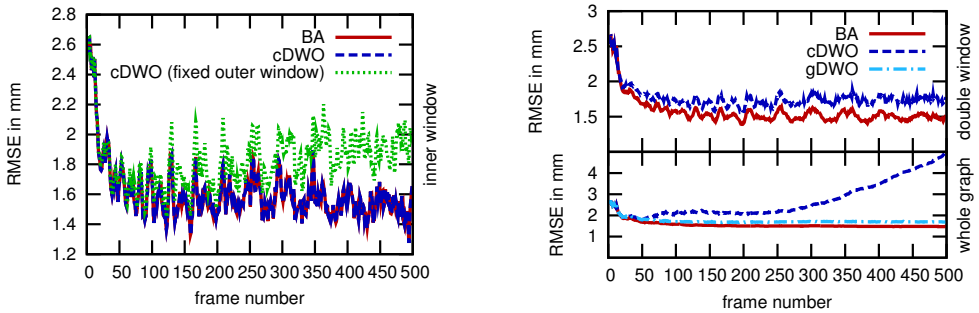
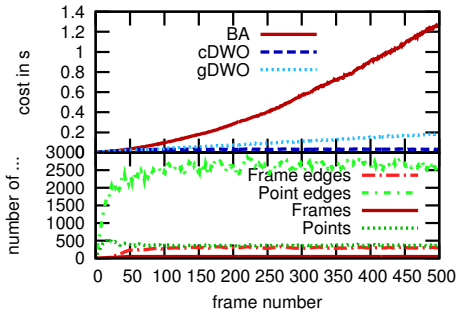


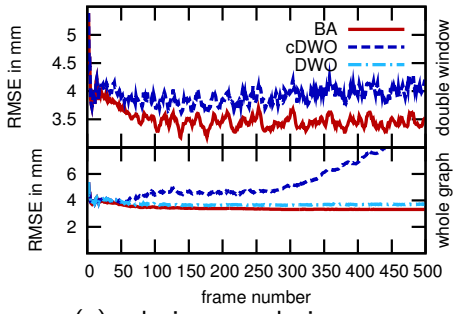
Figure 6.9: Spiral simulation scenario



(a) relative translation error, stereo



(b) computational cost



(c) relative translation error, monocular

Figure 6.10: Spiral simulation experiment. The plot shows averages over ten Monte Carlo trials.

both very loopy local motion and large scale exploration. We evaluate a combination of both of these in our first set of Monte-Carlo simulation experiments. Here the camera moves in a spiral (see Figure 6.9), and the trajectory consists of 500 keyframes. We assume a stereo camera model with focal length of 300, a baseline of $5cm$, a resolution of 640×480, Gaussian image noise of one pixels and perfect data association. We compare BA over all frames to our double-window optimisation. Both methods are implemented using the efficient state-of-the-art sparse graph optimizer **g²o** (Kümmerle et al., 2011a) and executed on a single core of an Intel(R) Core(TM) i7 960 desktop computer.

For each keyframe, we perform three iterations of joint structure and motion optimisation (BA or DWO). We apply two variants of the double window optimisation. The first variant (cDWO) is made to have strictly constant-time operation by restricting the inner window to 15 frames and the outer window to 50. In the second variant (gDWO) the outer window covers all remaining 485 frames and therefore allows global metric mapping.

In order to define an error measure, we have to remember that our scheme does not have a fixed global origin, and therefore comparing absolute poses is meaningless. Instead we follow the approach of Kümmerle et al. (2009) and define a relative error in terms of the relative differences $\Delta \mathtt{T}_{ij} := \mathtt{T}_i \mathtt{T}_j^{-1}$ between two absolute poses. In particular, we define the root mean square error (RMSE) over the difference of estimated and true relative translations,

$$\sqrt{\frac{1}{|\mathcal{E}|} \sum_{E_{ij} \in \mathcal{E}} \left(\Delta \mathbf{t}_{ij}^{[\text{est}]} - \Delta \mathbf{t}_{ij}^{[\text{true}]} \right)^2} , \tag{6.10}$$

with $\Delta \mathbf{t}_{ij}$ being the translational component of $\Delta \mathtt{T}_{ij}$. We analyse the RMSE at three different levels as shown in Figure 6.10(a). First, we consider the local error within the inner window (left). One can see that the constant time framework (cDWO) reaches the same accuracy as BA. Furthermore, we evaluated an adapted cDWO version where all frames in the outer window are fixed during optimisation. One can clearly see that the usage of such hard constraints can lead to inferior results. A second RMSE is computed at an intermediate level considering errors in both windows (top right). Once the 15th frame is passed cDWO slightly degrades from BA, but settles down quickly. Finally, we calculate a global error by considering all relative constraints (bottom right). Here, gDWO stabilises close to BA, while cDWO is clearly inferior since it only ensures accuracy within the double window.

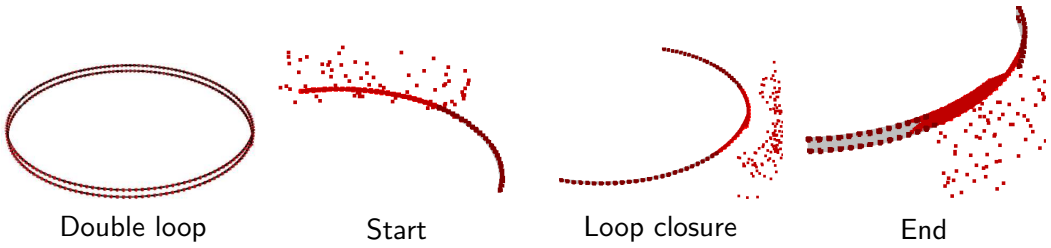
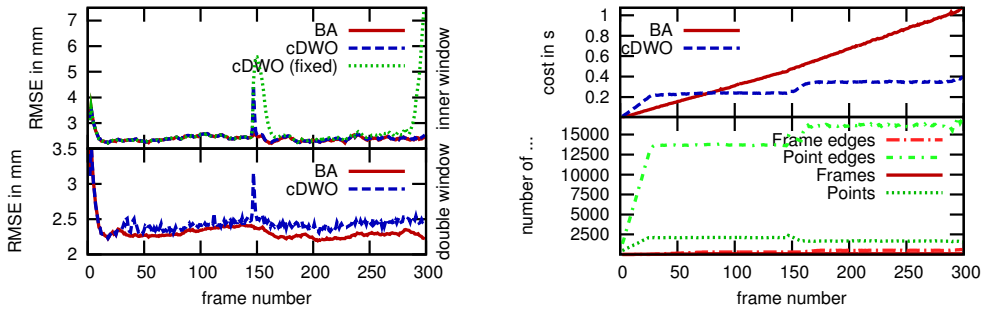


Figure 6.11: Double loop simulation scenario: At the start, the most recent 25 frames lie within the inner window, while the outer window is dragged behind. At loop closure, the inner window is at the centre, while the outer window extends in both directions.



(a) Translation error (b) Computational cost

Figure 6.12: Double loop experiment. The plot shows averages over ten Monte Carlo trials

Figure 6.10(b) illustrates the computational cost for all three methods (BA, cDWO, gDWO). The constant computation times of cDWO can be well understood by studying the number of frames, points, point-to-frame constraints and frame-to-frame constraints used within the optimisation windows (bottom). We performed a comparable simulation experiment using a monocular camera. The RMSE is converted into a scale-invariant version by normalising the translation vectors $\Delta\mathbf{t}_{ij}$ to length one. The corresponding accuracy plots are shown in Figure 6.10(c), forming a similar pattern as in stereo SLAM.

A second set of monocular simulation experiments is performed in order to demonstrate that our double window framework can deal with large loops and scale drift in a constant time fashion. The motion trajectory goes around a large loop twice as shown in Figure 6.11. The corresponding accuracy and cost are shown in Figure 6.12. Here, we have chosen an inner window size of 30 and an outer window size of 100. Note that the computational cost of cDWO slightly increases at loop

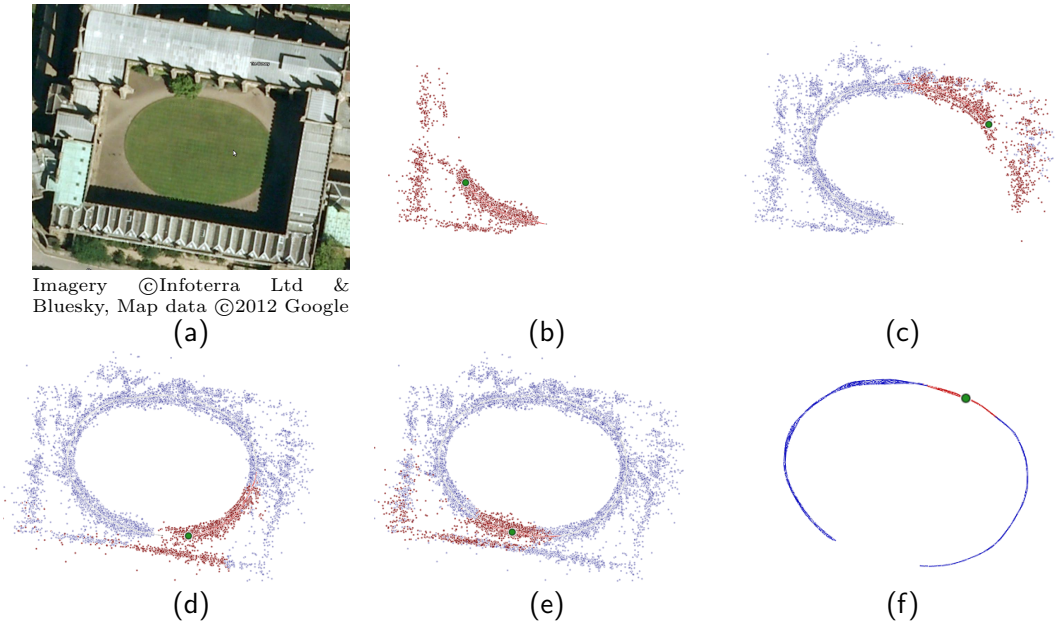


Figure 6.13: Stereo experiment, New College. The inner window is visualised in red; the outer window is visualised in blue (b-f).

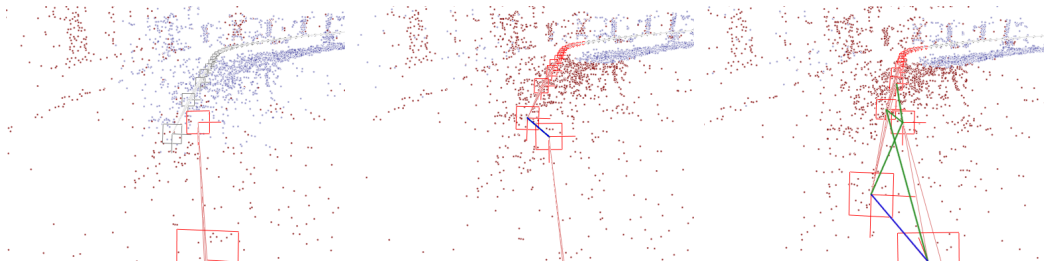
closure (frame 150), simply because the number of visible point-to-frame constraints increases as can be seen in Figure 6.12(b).

6.5.2 Real-image experiments

To further evaluate the DWO back-end, we have employed a range of real-image simulation experiments using stereo cameras, monocular cameras and an RGB-D camera. The experiments were performed on desktop computer with an Intel(R) Core(TM) 2 Duo CPU with 2.66 GHz and a NVIDIA 580 GTX (used for stereo front-end).

Stereo SLAM

We fully integrated the stereo front-end with DWO and the appearance based loop closure module. Each of the three modules is running in a separate thread. In order to be robust and efficient for large scale scenarios, we used the improved point parametrisation using anchored inverse depth features. The complete source code



(a) Right before (b) At large-scale loop closure (c) After

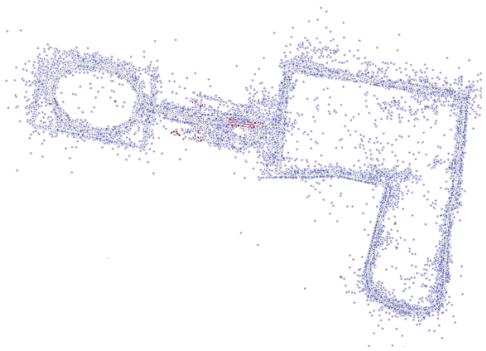
Figure 6.14: Large scale loop closure in stereo SLAM: At the loop closure event (b), the active (red) region expands. Afterwards (c), local registration is performed by means of metric loop closures (green).

is available online.[9] We performed several large scale SLAM experiments using the New College dataset[10] of Smith et al. (2009). It was recorded on and around the campus of the New College in Oxford by teleoperating a Segway robot which was equipped with several sensors including a Bumblebee stereo camera. The stereo images have a resolution of 512×384 and were recorded at 20 fps. Figure 6.13(a) shows an image of the New College quadrangle.

In a first experiment, we evaluated our stereo SLAM framework while the robot performs one and a half loops of the courtyard. We have chosen an outer window size ($|\mathcal{W}_1| = 25, |\mathcal{W}_2| = 175$) large enough to cover the whole loop. In this constant time setting, the runtime is with 25-30 fps faster than real-time. In the beginning, Figure 6.13(b), all frames are within the inner window. Figure 6.13(b) illustrates how the outer window is 'dragged behind' the inner window while the robot explores the environment. We visualise points in the inner window as well as in the outer window. Note that points in the outer window are not optimised over. However, we can visualise them efficiently since they are anchored relative to the optimised poses in the outer window. Figure 6.13(d) and Figure 6.14(a) illustrate the situation right before loop closure when the robot has traveled 115m. At this point, the SLAM graph consists of almost 13000 points and 149 keyframes which are inter-connected by over 400 edges. Most of the local registration was performed when new key frames are added; less than 10% are metric loop closure edges. At the appearance-based loop closure event (Figure 6.13(e) and Figure 6.14(b)), both ends of the graph get

[9] https://github.com/strasdat/ScaViSLAM
[10] http://www.robots.ox.ac.uk/NewCollegeData/

(a) Aerial image of the New College



(b) Metric point map

Figure 6.15: Large-scale experiment with 1.8 km trajectory. (a) shows an aerial photograph of the environment. The robot performed five loops of the quadrangle (top left) and two larger loops in the park (right). (b) shows the metric map consisting of over 2000 keyframes and over 200.000 points.

connected, and the inner window automatically expands so that it shapes around the current pose. The DWO framework is designed in such a way that appearance-based loop closure are rarely triggered — only to close large loops. Figure 6.14(c) shows how, after the appearance-based loop closure, the graph of robot poses become interconnected due to metric loop closure events and further local registration. When the 200th keyframe is added to the graph, the robot traveled for 155m. The SLAM graph consists now of over 17000 points and 812 edges — 110 of them are metric loop closures; no new appearance-based loop closure where triggered. After the robot had performed one and a half loops, the total number of keyframes exceeded the size of the double window, so that the outer window formed a stabilising periphery around the inner window (Figure 6.13(f)).

In a second experiment, we used a longer image sequence where the robot travelled for about 25 minutes and covers a distance of 1.8 km. In particular, the robot performed five loops on the courtyard of the campus and two larger loops in the park close-by (Figure 6.15(a)). The whole trajectory consists of over 2000 keyframes. A qualitative evaluation of this constant-time SLAM experiment is presented in Figure 6.16 and 6.17. Altogether, the appearance based loop closure place recognition module detected only 15 loops closures (Figure 6.18), while the vast majority of relative constraints were introduced due to local registration and metric loop closures. The efficiency of the appearance-based loop closure module is reflected by the

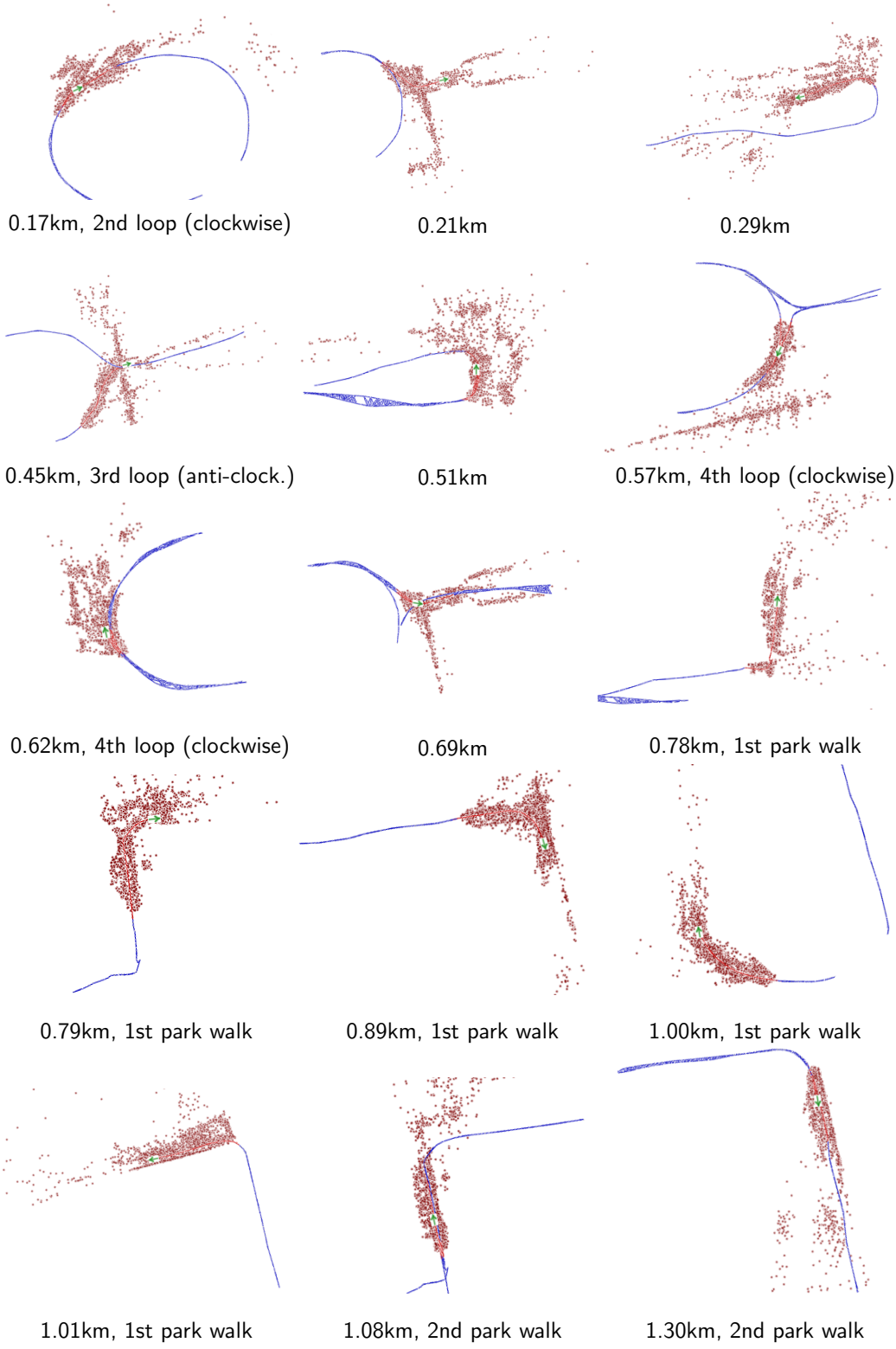


Figure 6.16: Constant-time, large scale stereo experiment. Top views of double window are shown over time (1-15).

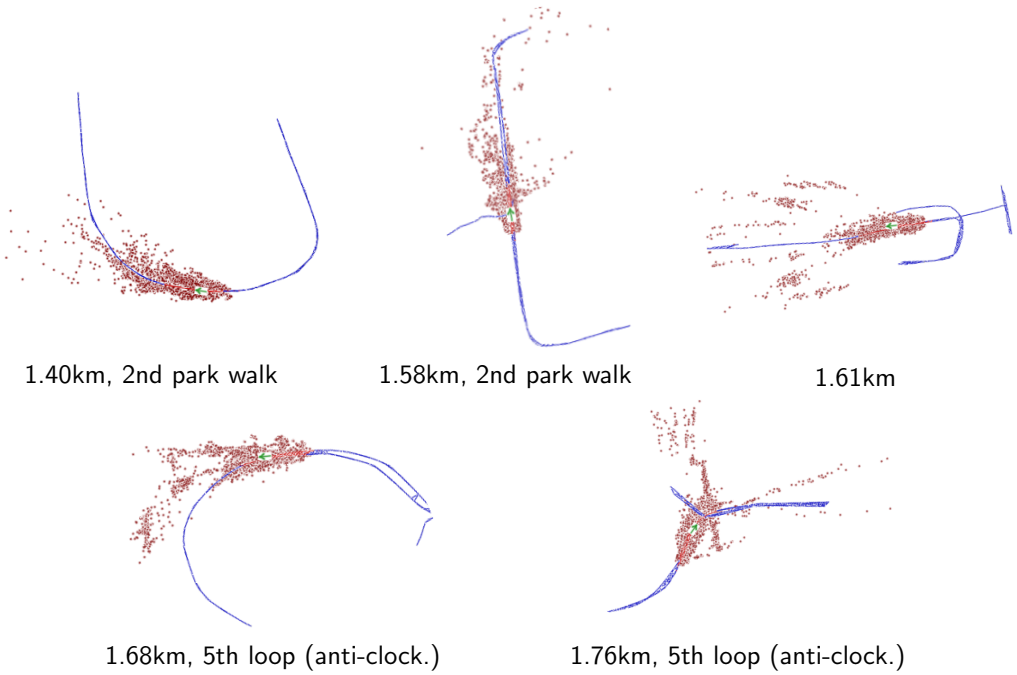


Figure 6.17: Constant-time, large scale stereo experiment (16-20).

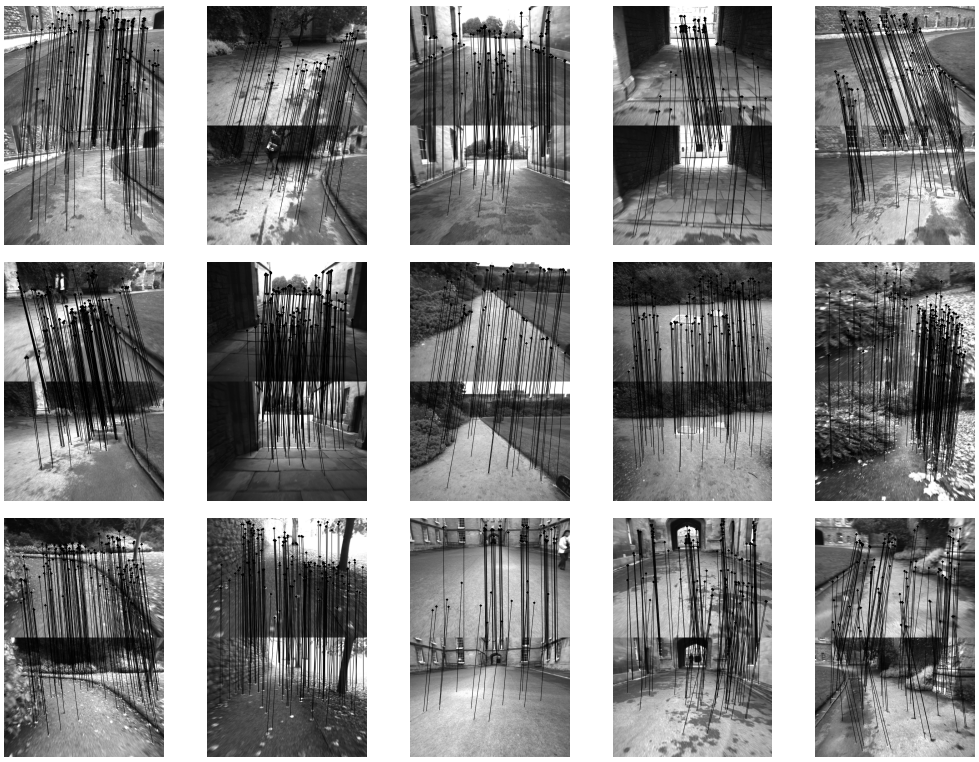


Figure 6.18: All appearance-based loop closures of the 1.8km experiment.

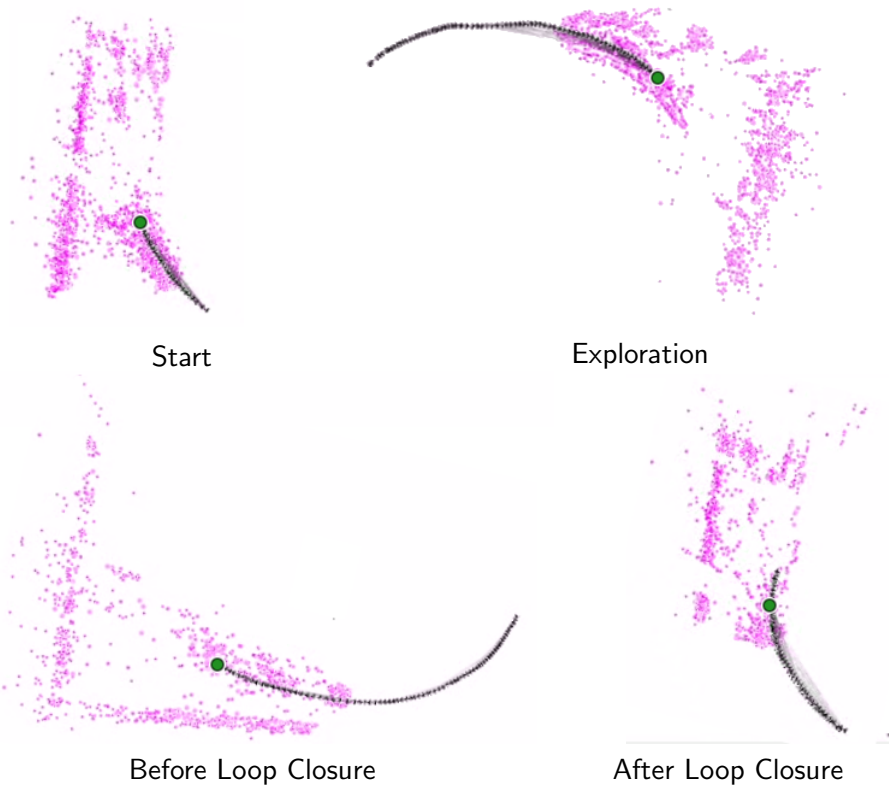


Figure 6.19: Monocular large-scale SLAM evaluated on the New College dataset in constant time (17 FPS)

fact that it only used approximately one third of its computational resources. In a final experiment, we increased the outer window enough so that the whole area was covered by the double window. Nevertheless, the whole framework still run in near real-time at 15-20 fps. Figure 6.15(b) shows the metric map of over 200,000 points.

Monocular SLAM

We tested our monocular SLAM framework (adapted PTAM tracker plus DWO back-end) on two different image sequences. The first one is the loop on campus of the New College. For the monocular experiment, we only used the images of the left camera of the stereo pair. We have chosen a strictly constant time setting, such that the double window only covers approximately a third of the loop. In particular we chosen an inner window of 20 keyframes and an outer window of 35.

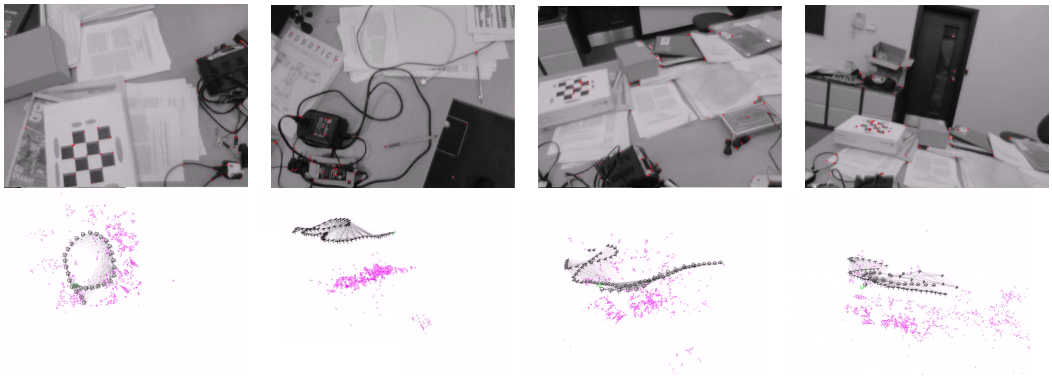


Figure 6.20: Loopy browsing motion plus exploration in office environment (monocular SLAM)

Qualitative results are shown in Figure 6.19. At loop closure, a scale drift of 6% is detected and both ends of the graph are attached appropriately. The whole system runs at 17 FPS — applying the tracker and the optimisation alternately in a single thread. In a second monocular experiment, we demonstrate that our back-end can deal with loopy browsing motion (which is the speciality of PTAM), but also with rapid exploration. Snap shots of the sequence are shown in Figure 6.20. Here, the camera is browsing over the desk, and rapidly explores in one direction. Then it returns to the origin and zooms out.

6.5.3 RGB-D SLAM

Our visual SLAM framework can be also used with RGB-D cameras that have become popular very recently. RGB-D cameras are devices with measure for each pixel not only intensities (RGB), but also depth (D). We used a device from PrimeSense which calculates a dense 3D cloud using structured light and is largely identical to Microsoft's Kinect. It outputs an RGB image together with a registered 3D point cloud. As a first step, we transform the 3D point cloud into a disparity image registered to the RGB image. By doing so, we can employ a stereo front-end such as the one described above.[11]

Figure 6.21 illustrates loopy browsing motion using the RGB-D camera. The tracking and optimisation solely depend on sparse feature matching (a). However,

[11] Indeed, the RGB-D experiments were performed using an older iteration of the stereo front-end described in Strasdat et al. (2011).

(a) Sparse point map (b) Dense 3D model

Figure 6.21: Loopy browsing motion in office environment, RGB-D

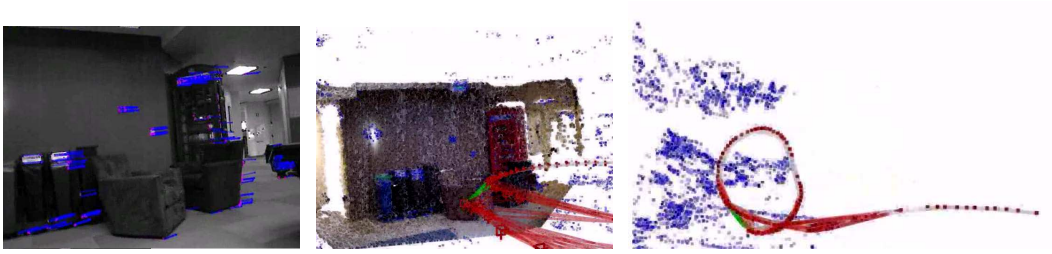


Figure 6.22: SLAM using a RGB-D camera on a wheeled robot.

the dense point clouds can be registered to the optimised frames and used to construct a dense environment model cheaply (b). Also, we attached the RGB-D camera to a wheeled robot and mapped an indoor environment (Figure 6.22),

Finally, we demonstrate how our framework can be used to create dense object models (see Figure 6.23). An object is placed on a turntable and observed by a static RGB-D camera. We need to create an image mask which only covers the object. First, we remove the background by rejecting all pixels with a depth greater than a particular threshold. Second, we detect the ground plane and only accept pixels whose corresponding 3D points are significantly above it.[12] Thirdly, pixels on the object boundary are smeared, since they mix colour values of the object with colour values of the background. Thus, we perform *erosion* operations on the object mask in order to make sure that pixels at the object boundary are rejected. Note that the loop is detected using a metric loop closure.

A number of videos are available online which illustrate our simulation and real-image experiments[13].

[12] Thanks to Suat Gedikli for providing the preprocessed data set.
[13] http://www.doc.ic.ac.uk/~strasdat/website/php/thesis

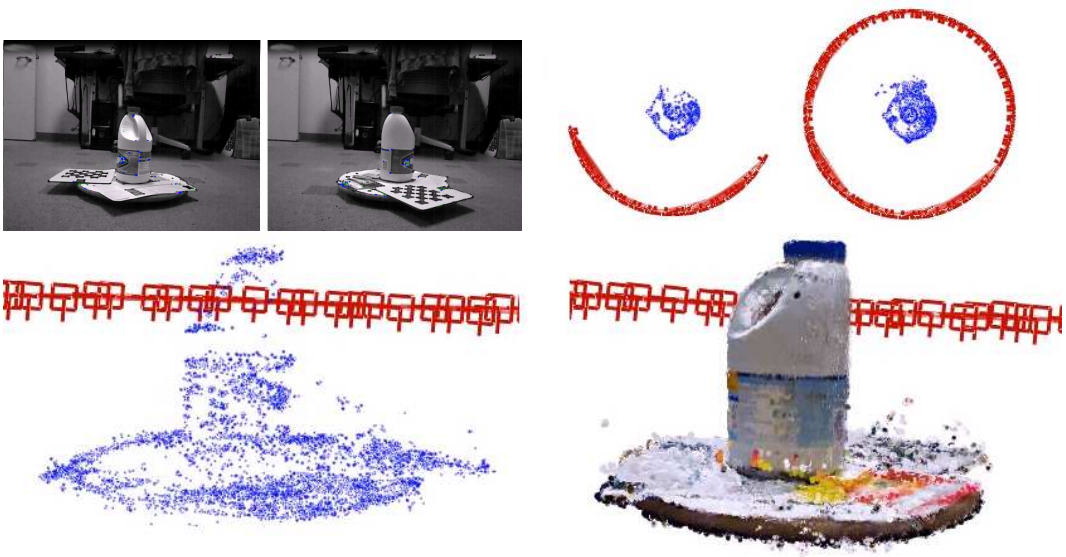


Figure 6.23: Dense object models using an RGB-D camera. Bottom left: Sparse points clouds are used for SLAM. Bottom right: RGB measurements are overlaid to create a dense object model.

6.6 Discussion and Summary

We have presented a novel framework for visual SLAM, which is unique in being able smoothly to cope with both detailed, loopy browsing, and rapid large-scale exploration in constant time, attaining comparable results to bundle adjustment locally. The whole map is represented using a graph of keyframes and points, while the optimisation is performed in a double window. We tested this double window optimisation exhaustively on a set of synthetic as well as real image experiments using data from monocular, stereo and RGB-D cameras.

Furthermore, we developed a novel stereo front-end, which combines dense 3D Lucas-Kanade tracking with a guided search of sparse 3D points. While the dense tracker ensures an accurate incremental ego motion estimate, local metric consistency is achieved by means of the optimised sparse point cloud.

Finally, we presented a mechanism for local registration using top-down feature search and metric loop closures. This enables the uniform treatment of repetitive local browsing and large loop mapping. As opposed to previous approaches, which tried to detect as many appearance-based loop closures as possible, we only aim to

detect them between places which are topologically distant and rely on top-down metric loop closures otherwise. This offers a computational advantage. While an appearance-based loop hypothesis needs to be evaluated using a potentially expensive validation scheme, metric loop closures only require a guided search and are therefore more efficient.[14] In addition, our approach seems to reduce the risk of false positive loop closures. First, we can rely on a reluctant detection scheme for place recognition since our framework only requires that appearance-based detection is performed once per large loop. Second, metric loop closures rely on a strong geometric prior so that false positives are only conceivable for pathological cases (such as highly repetitive scene structure). We did not observe any false positives; though a quantitative evaluation is pending.

6.7 Bibliographic Remarks

Topological maps already were used in early robotic research. Mataric (1990) highlighted the advantages of graph representations over metric maps for autonomous robot navigation. Besides performing real-robot experiments, she performed a review of biological studies: "insects, animals, and people use cognitive maps as internal representations of spatial information. The maps have been shown to contain both topological and metric information."

In their biologically inspired RatSLAM, Milford et al. (2004) combined a weak Cartesian grid with a topological structure. They stressed that for topologically usable maps it is not necessary to propagate metric errors along loops. Milford & Wyeth (2008) adopted this approach to a single-camera 2D SLAM framework. Howard et al. (2004) discussed the benefits of *manifolds* — a local Euclidean, but global topological representation — for 2D SLAM. In particular, they represent a two-dimensional manifold by means of a set of partially overlapping patches. They illustrated how such a representation sidesteps the problem of mapping cross-overs and elaborated the capabilities of lazy loop closure corrections.

Sibley et al. (2009) introduced Relative Bundle Adjustment (RBA). Unlike Howard et al. (2004) and other submapping approaches, it relies on a *continuous* manifold

[14] Very efficient appearance-based loop closure techniques were presented recently such as the one of Galvez-Lopez & Tardós (2012) which relies on *binary* bag of words. Still, such approaches require a RanSaC-like validation.

representation. In RBA, points and poses are represented in a purely relative manner. In order to calculate measurement predictions, points are projected into the designated frame along the chain of relative poses. An active window strategy enables constant-time performance. Mei et al. (2009, 2010a) integrated RBA in their real-time stereo SLAM framework and demonstrated its outstanding capabilities on a number of large-scale indoor and outdoor datasets (including New College). However it is unclear how RBA performs for repetitive local browsing since, unlike DWO, RBA does not enforce metric consistency on loopy graph structure.

Holmes et al. (2009) introduced a framework for efficient monocular SLAM by combining PTAM with RBA. In particular they showed how the relative representation allows the SLAM problem to be split into two subtasks. The local motion can be estimated in an active window of relative poses in constant-time, while the whole map is optimised using global RBA in the background. This approach enables local browsing and rapid exploration. However, Holmes et al. (2009) did not tackle the problems of large-scale loop closures and scale drift.

Castle et al. (2011) presented a monocular framework for large-scale augmented reality applications by extending PTAM (Klein & Murray, 2007) to multiple maps. The spatial relations between the maps is ignored, and map switching is performed using appearance-based relocalisation.

Handling relocalisation and loop closures using appearance-based place recognition is now the de-facto standard for visual SLAM. It has been used in the frameworks of Konolige & Agrawal (2008), Eade & Drummond (2008), Mei et al. (2009), Lim et al. (2011), Pirker et al. (2011), Johannsson et al. (2012) and many others. An early system was the one of Dudek & Jugessur (2000). They represented visual features using a rotation-invariant polar representation and performed dimensionality reduction using PCA. Sivic & Zisserman (2003) highlighted the strong relation between visual recognition and text retrieval. In particular, they introduce the concept of *visual words* and applied techniques from computational linguistics such as the tf-idf statistic. In their stand-out system, Nister & Stewenius (2006) performed object recognition using a vocabulary tree. Due to its hierarchical structure, the framework is highly scalable and was tested using a database of tens of thousands of images. Angeli et al. (2008) presented an incremental method for appearance-based loop closure detection where the dictionary of visual words is created online. Cummins & Newman's FAB-MAP (2008; 2009) is highly scalable appearance-only

SLAM framework which relies on the co-occurance probability of visual words and was demonstrated on a extremely long trajectory of 1000km. Maddern et al. (2012) combined FAB-MAP with metric pose filtering to improve the recall of loop closures. Galvez-Lopez & Tardós's approach (2012) rely on binary visual words and a binary vocabulary tree which results in a very efficient detection scheme.

In previous work, the concept of co-visibility was exploited in order to organize the SLAM graph topology. Often, keyposes are connected along the chain of motion plus additional links due to appearance-based place recognition as in Konolige & Agrawal (2008), Mei et al. (2009) and Lim et al. (2011). Instead, our approach is related to Mei et al. (2010b) who discussed the usefulness of co-visibility graphs for appearance-based loop closing. Especially, they highlighted the structural differences between the co-visibility graph and the RBA graph. DWO takes a step forward by defining the topology of the SLAM graph purely based on co-visibility which allows the active generation of loop hypothesis along the graph of relative constraints using metric priors. Williams et al. (2009) compared map-to-map, image-to-map and image-image loop closure techniques for monocular SLAM and concluded that image-to-map techniques work best; metric loop closures fall into this category.

DWO is related to the stereo SLAM approach of Lim et al. (2011). They represented the map using a similar graph structure, but performed local adjustment and global optimisation independently. For the global optimisation, the whole map is divided into a set of disjoint segments. First local segments are optimised using BA. Then global consistency is enforced by treating the segments as rigid bodies. Unlike in DWO, the visual front-end relies on bottom-up KLT tracking so that no previous geometric reconstruction can be exploited once a known place is revisited. Instead, all loop closures are detected using appearance information.

Another relevant work is the recent monocular approach of Pirker et al. (2011). It also relies on a keypose graph and performs local BA using an active window strategy. Loop closures are detected using FAB-MAP and metric consistency is enforced globally. Pirker et al. followed the approach described in Chapter 5; pose graph optimisation is used to correct for rotation, translation and scale drift. Optimisation is performed only over the uncorrected loop, while the remaining keyposes remain fixed.

Johannsson et al. (2012) presented a large-scale SLAM framework for range im-

age devices. Similar to Konolige & Agrawal (2008), they combined bottom-up visual odometry, appearance-based place recognition and pose-graph optimisation. A reduced pose-graph representation is used which does not scale with the distant travelled, but only with the area of operation. Johannsson et al. evaluated their approach on a large-scale dataset recorded in a ten-floor building; the vertical motion of elevators is estimated using an IMU.

SLAM using RGB-D cameras became popular very recently. Henry et al. (2010) used a structured light camera from PrimeSense to build a dense 3D map. They used depth and visual information in an combined RanSaC and iterative closes point (ICP, Besl & McKay, 1992) approach. Global consistency is enforced using pose-graph optimisation. Newcombe et al. (2011a) presented a real-time system for dense tracking and mapping. They ignore the intensity measurements and only rely on depth data. All depth measurement are fused into a single implicit surface, while the motion is estimated using coarse to fine ICP. The approach is highly robust and accurate, but the area of operation is limited (small room size).

This chapter is partially based on Strasdat et al. (2011).

Chapter 7

Conclusion

In this thesis, we have tackled efficient visual SLAM by concentrating on real-time strategies for *locally accurate* and *globally consistent* estimation of scene structure and camera motion. Previous approaches have been thoroughly analysed and a number of new techniques have been presented. The main achievements are:

- A rigorous comparison of Gaussian filtering versus keyframe bundle adjustment (Chapter 4). The main result is: *Increasing the number of observations N increases the accuracy, while increasing the number of intermediate frames M only has a minor effect. Considering the cost of bundle adjustment (linear in N) to the cost of filtering (cubic in N), it becomes clear that bundle adjustment is the more efficient technique – especially if high accuracy is required.*

- A framework for monocular exploration based on keyframe sliding window bundle adjustment (Chapter 3). A novel and effective monocular tracker has been developed which combines bottom-up dense optical flow with top-down guided search. The unknown depth of newly initialised feature is inferred using a set of Gauss-Newton filters.

- An efficient technique for monocular loop closure correction based on novel pose-graph optimisation by taking *scale drift* into account (Chapter 5). Since monocular SLAM does not only drift in rotation and translation, but also in scale, this new technique outperforms previous approaches designed for

range-bearing sensors. Furthermore, we showed how the Jacobians of general pose-graph problems can be calculated efficiently using an n-order Campell-Baker-Haussdorf expansion.

- A novel and unified framework for constant-time visual SLAM using *double window optimisation* (Chapter 6). Accurate reconstruction are performed in the inner window (which approaches the local accuracy of bundle adjustment) while the outer window acts as a stabilising periphery. Inner window pose-point constraints and outer pose-pose relations are jointly optimised using a common cost term.

- An effective strategy for *local registration* using metric loop closures and active point search in the topological neighbourhood of the SLAM graph. This offers a unified and efficient solution for repetitive local browsing and place revisits after large loop closure. By relying on the graph topology, occlusion problems are handled implicitly.

- An effective strategy for uniform feature selection using quadtrees. A new traversal technique has been developed (Chapter 3) which is used for feature selection throughout the thesis.

- A new stereo front-end based on hybrid dense/sparse tracking (Chapter 3). This approach combines 3D Lukas Kanade tracking with active search and has proved to be efficient, accurate and robust.

- Extensive Monte Carlo evaluations using synthetic data (Chapters 4,5,6) as well as real-image experiments using monocular, stereo and structured light cameras (Chapters 3,5,6).

7.1 Discussion and Future Work

Most recent large-scale SLAM approaches consist of largely independent modules. Bottom-up visual trackers and purely appearance-based place recognisers do not rely on any input but raw camera images. This design is appealing since it allows an uncomplicated software architecture — the modules can be evaluated individually and parallel computing is enabled straightforwardly. On the other hand, top-down and model-based approaches require a higher integration of the individual components

which complicates the software design. Our work highlights that such a complication is worthwhile. Top-down active search and metric loop closures highly facilitate local registration — a requirement which enables accurate and continues large-scale mapping in real-time. The resulting double window framework (laid out in Chapter 6) is widely applicable to general-purpose SLAM tasks, but also offers a basis for future work. The following three areas are especially promising:

- The double window framework employs effective heuristics which allow revisits of previous places again and again. If one takes a step forward, the problem of 'lifelong mapping' (Biber & Duckett, 2005; Konolige & Bowman, 2009; Pirker et al., 2011; McDonald et al., 2011; Churchill & Newman, 2012) emerges. In order to solve lifelong mapping, several difficulties need to be tackled. First and foremost, the environment is prone to change over time. One could approach this problem by inferring which parts of the environment are persistent and which ones are transient. Our double window approach offers some mechanisms already to handle changing environments. When a known place is revisits, it aims to join overlapping map segments using local registration. If this attempt fails in certain parts of the trajectory, it is an indication that a scene variation occurred. However, such a classification is not straight-forward; for instance variation in lightning affects scene appearance as well. Furthermore, long term mapping brings additional problems. Under ideal condition, the application of SLAM in a bounded environment has a constant space complexity. However, if SLAM is performed for hours, days and beyond and if the environment keeps on changing, new data need to be added constantly. Novel algorithm and heuristics (such as the removal of outdated features, constraints or even whole map segments) are required in order to restrict the memory requirement and allow continuous real-time operation.

- Apart from modelling large static environments which gradually change over time, there is the open challenge of performing SLAM in *dynamic scenes*. There has been initial work on real-time multibody SLAM (e.g. Kundu et al., 2011), but more work has to be done. Even more challenging is the problem of *non-rigid* structure and motion estimation (Fayad et al., 2011).

- Very recently, systems for real-time dense tracking and mapping were pre-

sented (Newcombe et al., 2011b; Graber et al., 2011). These frameworks perform highly parallel computation on modern GPUs and produce scene models which are more descriptive than sparse point clouds. Their accurate outputs allow the decoupling of scene reconstruction and camera tracking in small areas of operation. For large scale mapping drift is unavoidable, but the (efficient) joint estimation of dense structure and motion is an open problem. We belief there is space for a hybrid sparse/dense SLAM framework where direct and dense methods are used for accurate local reconstruction while global consistency is enforced using a sparse optimisation approach.

Appendix A

Proofs and Formulae related to Lie Groups

A.1 Generators

Let $\mathbf{G}$ be a Lie group, $\mathfrak{g}$ be the corresponding Lie algebra and $\widehat{\cdot}$ its hat-operator. The generators of $\mathbf{G}$ can be calculated as $\mathtt{G}_k = \widehat{\mathbf{e}_k}$ with $\mathbf{e}_k$ being the kth Cartesian unit vector.

A.1.1 Generators of SE(3)

$$\mathtt{G}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathtt{G}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathtt{G}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\mathtt{G}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathtt{G}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathtt{G}_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.1}$$

A.1.2 Generators of Sim(3)

The generators of **Sim**(3) includes the ones of **SE**(3) plus:

$$\mathtt{G}_7 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \tag{A.2}$$

A.2 Adjoint Representations

A.2.1 Adjoint Map of SE(3)

The adjoint map of **SE**(3) is

$$\mathtt{Ad}_\mathtt{T} = \begin{bmatrix} \mathtt{R} & [\mathbf{t}]_\times \mathtt{R} \\ \mathtt{0} & \mathtt{R} \end{bmatrix} \tag{A.3}$$

since

$$\begin{bmatrix} \mathtt{R} & [\mathbf{t}]_\times \mathtt{R} \\ \mathtt{0} & \mathtt{R} \end{bmatrix} \cdot \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \mathtt{R}v + \mathbf{t} \times \mathtt{R}\boldsymbol{\omega} \\ \mathtt{R}\boldsymbol{\omega} \end{pmatrix} = \begin{bmatrix} [\mathtt{R}\boldsymbol{\omega}]_\times & \mathtt{R}\boldsymbol{v} - (\mathtt{R}\boldsymbol{\omega}) \times \mathbf{t} \\ \mathtt{0} & 1 \end{bmatrix}^\vee_{\mathfrak{se}(3)} \tag{A.4}$$

and

$$\begin{bmatrix} [\mathtt{R}\boldsymbol{\omega}]_\times & -[\mathtt{R}\boldsymbol{\omega}]_\times \mathbf{t} + \mathtt{R}\boldsymbol{v} \\ \mathtt{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathtt{0} & 1 \end{bmatrix} \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{v} \\ \mathtt{0} & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R}^\top & -\mathtt{R}^\top \mathbf{t} \\ \mathtt{0} & 1 \end{bmatrix} . \tag{A.5}$$

A.2.2 Adjoint Map of Sim(3)

The adjoint map of **Sim**(3) is

$$\mathtt{Ad}_\mathtt{S} = \begin{bmatrix} s\mathtt{R} & [\mathbf{t}]_\times \mathtt{R} & -\mathbf{t} \\ \mathtt{0}_{3\times 3} & \mathtt{R} & \mathbf{0} \\ \mathtt{0}_{1\times 3} & \mathtt{0}_{1\times 3} & 1 \end{bmatrix} \tag{A.6}$$

since

$$\begin{bmatrix} s\mathtt{R} & [\mathbf{t}]_\times \mathtt{R} & \mathbf{t} \\ \mathtt{0}_{3\times 3} & \mathtt{R} & \mathbf{0} \\ \mathtt{0}_{1\times 3} & \mathtt{0}_{1\times 3} & 1 \end{bmatrix} \cdot \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \\ \sigma \end{pmatrix} = \begin{pmatrix} s\mathtt{R}\boldsymbol{v} + [\mathbf{t}]_\times \mathtt{R}\boldsymbol{\omega} - \sigma\mathbf{t} \\ \mathtt{R}\boldsymbol{\omega} \\ \sigma \end{pmatrix} \tag{A.7}$$

$$= \begin{bmatrix} [\mathtt{R}\boldsymbol{\omega}]_\times + \sigma \mathtt{I} & s\mathtt{R}\boldsymbol{v} - [\mathtt{R}\boldsymbol{\omega}]_\times \mathbf{t} - \sigma\mathbf{t} \\ \mathtt{0} & 0 \end{bmatrix}^\vee_{\mathfrak{sim}(3)} \tag{A.8}$$

and

$$\begin{bmatrix} [\mathtt{R}\boldsymbol{\omega}]_\times + \sigma\mathtt{I} & -[\mathtt{R}\boldsymbol{\omega}]_\times \mathbf{t} - \sigma\mathbf{t} + s\mathtt{R}\boldsymbol{v} \\ \mathtt{0} & 0 \end{bmatrix} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathtt{0} & 1 \end{bmatrix} \begin{bmatrix} [\boldsymbol{\omega}]_\times + \sigma\mathtt{I} & \boldsymbol{v} \\ \mathtt{0} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{s}\mathtt{R}^\top & -\frac{1}{s}\mathtt{R}^\top \mathbf{t} \\ \mathtt{0} & 1 \end{bmatrix}. \tag{A.9}$$

A.3 Lie brackets

A.3.1 Derivation of the Lie Bracket from the Adjoint

Let $\mathbf{G}$ be a Lie group and $\mathfrak{g}$ be the corresponding tangent space. Furthermore, let us assume that $\mathtt{A}$ is a smooth path through the identity. Thus, we can assume that $\mathtt{A}(t) = \exp(t\mathtt{U})$ with $\mathtt{U}$ being a general tangent vector in $\mathfrak{g}$. If we differentiate the adjoint map $\mathrm{Adj}_{\mathtt{A}}(\mathtt{V}) = \mathtt{A}(t) \cdot \mathtt{V} \cdot \mathtt{A}(t)^{-1}$ with respect to $\mathtt{A}$, we end up with the Lie bracket:

$$\frac{\partial}{\partial t} \mathrm{Adj}_{\mathtt{A}(t)}(\mathtt{V}) \Big|_{t=0} = \frac{\partial}{\partial t} \left(\exp(t\mathtt{U}) \cdot \mathtt{V} \cdot (\exp(t\mathtt{U}))^{-1} \right) \Big|_{t=0} \tag{A.10}$$

$$\overset{(2.69)}{=} \frac{\partial}{\partial t} \left(\exp(t\mathtt{U}) \cdot \mathtt{V} \cdot \exp(-t\mathtt{U}) \right) \Big|_{t=0} \tag{A.11}$$

$$\overset{(2.64)}{=} \mathtt{U}\exp(\mathtt{0})\mathtt{V} - \mathtt{V}\exp(\mathtt{0})\mathtt{U} \tag{A.12}$$

$$\overset{(2.67)}{=} \mathtt{UV} - \mathtt{VU}\,. \tag{A.13}$$

In the following, we give close form solutions for the Lie brackets and its derivatives using the minimal vector representations for various relevant Lie groups.

A.3.2 Lie bracket and its Jacobian of SO(3)

Let $\boldsymbol{\omega}$, $\boldsymbol{\phi}$ be elements of the Lie algebra $\mathfrak{so}(3)$. Their Lie bracket is:

$$\begin{aligned}[\boldsymbol{\omega},\boldsymbol{\phi}]_{\mathfrak{so}(3)} &= (\widehat{\boldsymbol{\omega}}\widehat{\boldsymbol{\phi}} - \widehat{\boldsymbol{\phi}}\widehat{\boldsymbol{\omega}})^{\vee} = \begin{bmatrix} 0 & -\omega_1\phi_2 + \omega_2\phi_1 & -\omega_1\phi_3 + \omega_3\phi 1 \\ \omega_1\phi_2 - \omega_2\phi_1 & 0 & -\omega_2\phi_3 + \omega_3\phi_2 \\ \omega_1\phi_3 - \omega_3\phi_1 & \omega_2\phi_3 - \omega_3\phi_2 & 0 \end{bmatrix}^{\vee}_{\mathfrak{so}(3)} \\ &= \begin{pmatrix} \omega_2\phi_3 - \omega_3\phi_2 \\ \omega_3\phi_1 - \omega_1\phi 3 \\ \omega_1\phi_2 - \omega_2\phi_1 \end{pmatrix} = \boldsymbol{\omega} \times \boldsymbol{\phi} \ . \end{aligned} \tag{A.14}$$

We can calculate the partial derivative $\frac{\partial}{\partial \omega_i}(\boldsymbol{\omega} \times \boldsymbol{\phi}) = (\mathbf{e}_i \times \boldsymbol{\phi})$; for instance $\frac{\partial}{\partial \omega_1}(\boldsymbol{\omega} \times \boldsymbol{\phi}) = (0, -\phi_3, \phi_2)^{\top}$. Thus, we get

$$\frac{\partial}{\partial \boldsymbol{\omega}}(\boldsymbol{\omega} \times \boldsymbol{\phi}) = -[\boldsymbol{\phi}]_{\times} \ . \tag{A.15}$$

A.3.3 Lie bracket and its Jacobian of SE(3)

The Lie bracket for elements of the Lie algebra $\mathfrak{se}(3)$ is:

$$\left[\begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\phi} \end{pmatrix}\right]_{\mathfrak{se}(3)} = \begin{pmatrix} \boldsymbol{\omega} \times \boldsymbol{\tau} + \boldsymbol{\upsilon} \times \boldsymbol{\phi} \\ \boldsymbol{\omega} \times \boldsymbol{\phi} \end{pmatrix} \ . \tag{A.16}$$

The derivative wrt. the translational component is $\frac{\partial}{\partial \boldsymbol{\upsilon}}\left[\begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\phi} \end{pmatrix}\right]_{\mathfrak{se}(3)} = \begin{pmatrix} -[\boldsymbol{\phi}]_{\times} \\ \mathsf{0}_{3\times 3} \end{pmatrix}$ while the derivative wrt. the rotational component equals $\frac{\partial}{\partial \boldsymbol{\omega}}\left[\begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\phi} \end{pmatrix}\right]_{\mathfrak{se}(3)} = \begin{pmatrix} -[\boldsymbol{\tau}]_{\times} \\ -[\boldsymbol{\phi}]_{\times} \end{pmatrix}$. Thus, the full Jacobian is

$$\frac{\partial}{\partial \mathbf{u}}[\mathbf{u},\mathbf{v}]_{\mathfrak{se}(3)} = \begin{bmatrix} -[\boldsymbol{\phi}]_{\times} & -[\boldsymbol{\tau}]_{\times} \\ \mathsf{0}_{3\times 3} & -[\boldsymbol{\phi}]_{\times} \end{bmatrix} \quad \text{with} \quad \mathbf{u} = \begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\phi} \end{pmatrix} \ . \tag{A.17}$$

A.3.4 Lie bracket and its Jacobian of Sim(3)

Finally, we present the Lie bracket of the tangent space $\mathfrak{sim}(3)$:

$$\left[\begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \\ \sigma \end{pmatrix}, \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\phi} \\ \varsigma \end{pmatrix}\right]_{\mathfrak{sim}(3)} = \begin{pmatrix} \boldsymbol{\omega} \times \boldsymbol{\tau} + \boldsymbol{\upsilon} \times \boldsymbol{\phi} + \sigma\boldsymbol{\tau} - \varsigma\boldsymbol{\upsilon} \\ \boldsymbol{\omega} \times \boldsymbol{\phi} \\ 0 \end{pmatrix} \ . \tag{A.18}$$

Its Jacobian is:

$$\frac{\partial}{\partial \mathbf{u}}[\mathbf{u},\mathbf{v}]_{\mathfrak{sim}(3)} = \begin{bmatrix} -[\boldsymbol{\phi}]_\times - \varsigma \mathtt{I} & -[\boldsymbol{\tau}]_\times & \boldsymbol{\tau} \\ \mathtt{0}_{3\times 3} & -[\boldsymbol{\phi}]_\times & \mathbf{0} \\ \mathtt{0}_{1\times 3} & \mathtt{0}_{1\times 3} & 0 \end{bmatrix} \quad \text{with} \quad \mathbf{u} = \begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \\ \sigma \end{pmatrix} \quad , \quad \mathbf{v} = \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\phi} \\ \varsigma \end{pmatrix} . \tag{A.19}$$

A.4 The Campbell-Baker-Hausdorff Formula

Let $\mathbf{G}$ be a Lie group and $\mathtt{A}, \mathtt{B} \in \mathbf{G}$. It is of interest to ask what

$$\mathrm{CBH}(\mathtt{A},\mathtt{B}) := \log(\exp(\mathtt{A}) \cdot \exp(\mathtt{B})) \tag{A.20}$$

looks like. If $\mathtt{A}$ and $\mathtt{B}$ commute, it follows from (2.65) that $\mathrm{CBH}(\mathtt{A},\mathtt{B}) = \mathtt{A} + \mathtt{B}$:

$$\mathtt{AB} = \mathtt{BA} \quad \Rightarrow \log(\exp(\mathtt{A}) \cdot \exp(\mathtt{B})) = \mathtt{A} + \mathtt{B} \; . \tag{A.21}$$

For commutative Lie groups, multiplications in the Lie group is equivalent to addition in the tangent space. Thus, the *global* structure of a commutative Lie group is covered by its tangent space. For instance for $\mathbf{SO}(2)$ it holds that

$$\gamma = \alpha + \beta \quad \Leftrightarrow \quad \mathtt{R}(\gamma) = \mathtt{R}(\alpha)\mathtt{R}(\beta) \; . \tag{A.22}$$

For general Lie groups it can be shown (e.g. Rossmann, 2002, pp.22) that

$$\mathrm{CBH}(\mathtt{X},\mathtt{Y}) = \sum_{k=0}^{\infty} \frac{(-1)^{k-1}}{k} \sum_{(r_k,s_k)\in\mathbb{N}^2, r_k+s_k\geq 1} \frac{\mathtt{X}^{r_1}\mathtt{Y}^{s_1}\cdots\mathtt{X}^{r_k}\mathtt{Y}^{s_k}}{r_1!s_1!\cdots r_k!s_k!} \; . \tag{A.23}$$

This bulky formula has no closed form solution. The first three terms are:

$$T_1 = \mathtt{X} + \mathtt{Y} \; , \tag{A.24}$$

$$T_2 = \frac{1}{2}[\mathtt{X},\mathtt{Y}] \; , \tag{A.25}$$

$$T_3 = \frac{1}{12}\left([\mathtt{X},[\mathtt{X},\mathtt{Y}]] - [\mathtt{Y},[\mathtt{X},\mathtt{Y}]]\right) \; . \tag{A.26}$$

All higher-order terms can be expressed using Lie brackets over $\mathtt{X}$ and $\mathtt{Y}$ (Stillwell, 2008, p.153). We can define a variant of the Campbell-Baker-Hausdorff formula

$$\mathrm{cbh} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m, \quad \mathrm{cbh}(\mathbf{x},\mathbf{y}) = \log(\exp(\widehat{\mathbf{x}}) \cdot \exp(\widehat{\mathbf{y}}))^\vee \tag{A.27}$$

where the Lie algebra elements are minimal vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$.

A.5 Exponential Map onto Sim(3)

The exponential map $\exp : \mathfrak{sim}(3) \to \mathbf{Sim}(3)$ is given by

$$\exp\begin{pmatrix} [\boldsymbol{\omega}]_\times + \sigma \mathtt{I}_{3\times 3} & \boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 0 \end{pmatrix} = \begin{pmatrix} e^\sigma \exp([\boldsymbol{\omega}]_\times) & \mathtt{W}\boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 0 \end{pmatrix} \tag{A.28}$$

with

$$\mathtt{W} = C\mathtt{I} + \frac{A\sigma + (1-B)\theta}{\sigma^2+\theta^2}\left(\frac{[\boldsymbol{\omega}]_\times}{\theta}\right) + \left(C - \frac{(B-1)\sigma + A\theta}{\sigma^2+\theta^2}\right)\left(\frac{[\boldsymbol{\omega}]_\times}{\theta}\right)^2, \tag{A.29}$$

$A = e^\sigma \sin(\theta)$, $B = e^\sigma \cos(\theta)$, $C = \frac{e^\sigma - 1}{\sigma}$ and $\theta = ||\boldsymbol{\omega}||_2$.

Proof

In order to derive the exponential map for $\mathbf{Sim}(3)$, we employ lemma (2.98). It follows immediately that

$$\exp\begin{pmatrix} [\boldsymbol{\omega}]_\times + \sigma \mathtt{I}_{3\times 3} & \boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 0 \end{pmatrix} = \begin{pmatrix} \exp([\boldsymbol{\omega}]_\times + \sigma \mathtt{I}_{3\times 3}) & \mathtt{W}\boldsymbol{\upsilon} \\ \mathtt{0}_{1\times 3} & 0 \end{pmatrix}, \tag{A.30}$$

with $\mathtt{W} = \sum_{k=0}^{\infty} \frac{\mathtt{X}^k}{(k+1)!}$ and $\mathtt{X} = [\boldsymbol{\omega}]_\times + \sigma\mathtt{I}$. Furthermore, $\exp([\boldsymbol{\omega}]_\times + \sigma \mathtt{I}_{3\times 3}) = e^\sigma \exp([\boldsymbol{\omega}]_\times)$. Thus, it remains to show that

$$\mathtt{W} = C\mathtt{I} + \frac{A\sigma + (1-B)\theta}{\sigma^2+\theta^2}\left(\frac{[\boldsymbol{\omega}]_\times}{\theta}\right) + \left(C - \frac{(B-1)\sigma + A\theta}{\sigma^2+\theta^2}\right)\left(\frac{[\boldsymbol{\omega}]_\times}{\theta}\right)^2. \tag{A.31}$$

Let $\Omega := \frac{1}{\theta}[\boldsymbol{\omega}]_\times$. It holds that $[\boldsymbol{\omega}]_\times = \theta\Omega$, $[\boldsymbol{\omega}]_\times^2 = \theta^2\Omega^2$, and $[\boldsymbol{\omega}]_\times^3 = -\theta^3\Omega$ since $\Omega^3 = -\Omega$ (Gallier, 2011, pp.471). Thus, in general we get for $k \in \mathbb{N}$:

$$\begin{aligned} [\boldsymbol{\omega}]_\times^{4k+1} &= \theta^{4k+1}\Omega, & [\boldsymbol{\omega}]_\times^{4k+2} &= \theta^{4k+2}\Omega^2, \\ [\boldsymbol{\omega}]_\times^{4k+3} &= -\theta^{4k+3}\Omega, & \text{and} \quad [\boldsymbol{\omega}]_\times^{4k+4} &= -\theta^{4k+4}\Omega^2. \end{aligned} \tag{A.32}$$

Now we can calculate the first few terms of the sequence $\mathtt{X}^k$

$$\begin{aligned} \mathtt{X} &= \theta\Omega + (\sigma-\sigma)\Omega^2 + \sigma\mathtt{I}, \\ \mathtt{X}^2 &= (2\sigma\theta)\Omega + (\sigma^2 - (\sigma^2-\theta^2))\Omega^2 + \sigma^2\mathtt{I}, \\ \mathtt{X}^3 &= (3\sigma^2\theta - \theta^3))\Omega + (\sigma^3 - (\sigma^3 - 3\sigma\theta^2)\Omega^2 + \sigma^3\mathtt{I}, \\ \mathtt{X}^4 &= (4\sigma^3\theta - 4\sigma\theta^3)\Omega + (\sigma^4 - (\sigma^4 - 4\sigma^2\theta^2 + \theta^4))\Omega^2 + \sigma^4\mathtt{I}, \\ \mathtt{X}^5 &= (5\sigma^4\theta - 10\sigma^2\theta^3 + \theta^5)\Omega + (\sigma^5 - (\sigma^5 - 10\sigma^3\theta^2 + 5\sigma\theta^3))\Omega^2 + \sigma^5\mathtt{I}, \end{aligned} \tag{A.33}$$

and so on. Note that the kth term consists of binomial coefficients of the order k. Using a proof by induction, one can show that the general pattern is:

$$\mathtt{X}^k = \sigma^k \mathtt{I} + \Phi_{\text{odd}}(\theta, \sigma, k)\Omega + \left(\sigma^k - \Phi_{\text{even}}(\theta, \sigma, k)\right)\Omega^2 \tag{A.34}$$

$$\text{with} \quad \Phi_{\text{odd}}(\theta, \sigma, k) = \left(\sum_{j=0}^{\lfloor \frac{k}{2} \rfloor} (-1)^j \binom{k}{2j+1} \sigma^{k-(2j+1)} \theta^{2j+1}\right) , \tag{A.35}$$

$$\Phi_{\text{even}}(\theta, \sigma, k) = \left(\sum_{j=0}^{\lfloor \frac{k}{2} \rfloor} (-1)^j \binom{k}{2j} \sigma^{k-2j} \theta^{2j}\right) , \tag{A.36}$$

and $\lfloor \cdot \rfloor$ being the floor operator. Hence, we get for $\mathtt{W} = \sum_{k=0}^{\infty} \frac{\mathtt{X}^k}{(k+1)!}$,

$$\mathtt{W} = \sum_{k=0}^{\infty} \frac{\sigma^k}{(k+1)!}\mathtt{I} + \sum_{k=0}^{\infty} \frac{\Phi_{\text{odd}}(\theta, \sigma, k)}{(k+1)!}\Omega + \left(\sum_{k=0}^{\infty} \frac{\sigma^k}{(k+1)!} - \sum_{k=0}^{\infty} \frac{\Phi_{\text{even}}(\theta, \sigma, k)}{(k+1)!}\right)\Omega^2 . \tag{A.37}$$

Since $\Omega = \frac{1}{\theta}[\boldsymbol{\omega}]_\times$ and

$$\sum_{k=0}^{\infty} \frac{x^k}{(k+1)!} = \frac{\sum_{k=0}^{\infty} \frac{x^k}{k!} - 1}{x} = \frac{e^x - 1}{x} , \tag{A.38}$$

we receive:

$$\mathtt{W} = C\mathtt{I} + \sum_{k=0}^{\infty} \frac{\Phi_{\text{odd}}(\theta, \sigma, k)}{(k+1)!} \left(\frac{[\boldsymbol{\omega}]_\times}{\theta}\right) + \left(C - \sum_{k=0}^{\infty} \frac{\Phi_{\text{even}}(\theta, \sigma, k)}{(k+1)!}\right) \left(\frac{[\boldsymbol{\omega}]_\times}{\theta}\right)^2 \tag{A.39}$$

with $C = \left(\frac{e^\sigma - 1}{\sigma}\right)$. Comparing this with proposition (A.29), it remains to show that $\sum_{k=0}^{\infty} \frac{\Phi_{\text{odd}}(\theta,\sigma,k)}{(k+1)!} = \frac{A\sigma + (1-B)\theta}{\sigma^2+\theta^2}$ and $\sum_{k=0}^{\infty} \frac{\Phi_{\text{even}}(\theta,\sigma,k)}{(k+1)!} = \frac{(B-1)\sigma + A\theta}{\sigma^2+\theta^2}$. Thus, we can

finish the proof with

$$
\begin{aligned}
& \sum_{k=0}^{\infty} \frac{\Phi_{\text{odd}}(\theta,\sigma,k)}{(k+1)!} \\
= \quad & \sum_{k=0}^{\infty} \frac{1}{(k+1)!} \sum_{j=0}^{\lfloor \frac{k}{2} \rfloor} (-1)^j \binom{k}{2j+1} \sigma^{k-(2j+1)} \theta^{2j+1} \\
= \quad & \frac{\theta}{2!} + \frac{2\sigma\theta}{3!} + \frac{3\sigma^2\theta - \theta^3}{4!} + \frac{4\sigma^3\theta - 4\sigma\theta^3}{5!} + \dots \\
= \quad & \frac{1}{2i} \left[1 + \frac{\sigma + i\theta}{2!} + \frac{\sigma^2 + 2i\sigma\theta - \theta^2}{3!} + \frac{\sigma^3 + 3i\sigma^2\theta - 3\sigma\theta^2 - i\theta^3}{4!} + \dots \right. \\
& \left. - \left(1 + \frac{\sigma - i\theta}{2!} + \frac{\sigma^2 - 2i\sigma\theta - \theta^2}{3!} + \frac{\sigma^3 - 3i\sigma^2\theta - 3\sigma\theta^2 + i\theta^3}{4!} + \dots \right) \right] \\
= \quad & \frac{1}{2i} \left(\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\sigma + i\theta)^n - \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\sigma - i\theta)^n \right) \\
\overset{(A.38)}{=} \quad & \frac{1}{2i} \left(\frac{e^{\sigma+i\theta} - 1}{\sigma + i\theta} - \frac{e^{\sigma-i\theta} - 1}{\sigma - i\theta} \right) \\
= \quad & \frac{e^\sigma \sigma \sin(\theta) + \theta - e^\sigma \theta \cos(\theta)}{\sigma^2 + \theta^2} \\
= \quad & \frac{A\sigma + (1-B)\theta}{\sigma^2 + \theta^2} \,,
\end{aligned}
$$

and

$$
\begin{aligned}
& \sum_{k=0}^{\infty} \frac{\Phi_{\text{even}}(\theta,\sigma,k)}{(k+1)!} \\
= \quad & \sum_{k=0}^{\infty} \frac{1}{(k+1)!} \sum_{j=0}^{\lfloor \frac{k}{2} \rfloor} (-1)^j \binom{k}{2j} \sigma^{k-2j} \theta^{2j} \\
= \quad & 1 + \frac{\sigma}{2!} + \frac{\sigma^2 + \theta^2}{3!} + \frac{\sigma^3 - \sigma\theta^2}{4!} + \frac{4\sigma^4 - 6\sigma^2\theta^2 + \theta^2}{5!} + \cdots \\
= \quad & \frac{1}{2} \left[1 + \frac{\sigma + i\theta}{2!} + \frac{\sigma^2 + 2i\sigma\theta - \theta^2}{3!} + \frac{\sigma^3 + 3i\sigma^2\theta - 3\sigma\theta^2 - i\theta^3}{4!} + \dots \right. \\
& \left. + \left(1 + \frac{\sigma - i\theta}{2!} + \frac{\sigma^2 - 2i\sigma\theta - \theta^2}{3!} + \frac{\sigma^3 - 3i\sigma^2\theta - 3\sigma\theta^2 + i\theta^3}{4!} + \dots \right) \right] \\
= \quad & \frac{1}{2} \left(\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\sigma + i\theta)^n + \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\sigma - i\theta)^n \right) \\
\overset{(A.38)}{=} \quad & \frac{1}{2} \left(\frac{e^{\sigma+i\theta} - 1}{\sigma + i\theta} + \frac{e^{\sigma-i\theta} - 1}{\sigma - i\theta} \right) \\
= \quad & \frac{e^\sigma \sigma \cos(\theta) - \sigma + e^\sigma \theta \sin(\theta)}{\sigma^2 + \theta^2} \\
= \quad & \frac{(B-1)\sigma + A\theta}{\sigma^2 + \theta^2} \,. \quad \square
\end{aligned}
$$

A.6 Derivative of the Lie Logarithm

Let $\mathbf{G}$ be a Lie group, $\mathtt{A}, \mathtt{B} \in \mathbf{G}$, $\mathtt{Ad}_{\mathtt{A}}$ the adjoint of $\mathtt{A}$ and $\widehat{\mathbf{d}} = \log(\mathtt{AB})$.

$$
\begin{aligned}
\mathtt{J}_{\mathtt{A}}^{\mathtt{B}} &:= \frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{A}\exp(\boldsymbol{\epsilon})\mathtt{B})^{\vee}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}} && \text{(A.40)}\\
&\overset{(2.78)}{=} \frac{\partial}{\partial \boldsymbol{\epsilon}} \log\left(\exp(\widehat{\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}})\mathtt{AB}\right)^{\vee}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}} && \text{(A.41)}\\
&= \frac{\partial}{\partial \boldsymbol{\epsilon}} \log\left(\exp(\widehat{\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}})\exp(\widehat{\mathbf{d}})\right)^{\vee}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}} && \text{(A.42)}\\
&= \frac{\partial}{\partial \boldsymbol{\epsilon}} \mathrm{cbh}(\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}, \mathbf{d})\Big|_{\boldsymbol{\epsilon}=\mathbf{0}} && \text{(A.43)}\\
&\overset{(*)}{\approx} \frac{\partial}{\partial \boldsymbol{\epsilon}} \left(\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon} + \frac{1}{2}[\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}, \mathbf{d}] + \frac{1}{12}([\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}, [\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}, \mathbf{d}]] + [\mathbf{d}, [\mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}, \mathbf{d}]])\right)_{\boldsymbol{\epsilon}=\mathbf{0}} && \text{(A.44)}\\
&= \left(\frac{\partial \mathbf{y}}{\partial \mathbf{y}} + \frac{1}{2}\cdot\frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\Big|_{\mathbf{y}=\mathbf{0}}\right. && \text{(A.45)}\\
&\quad \left. + \frac{1}{12}\left(\frac{\partial[\mathbf{y}, [\mathbf{y}, \mathbf{d}]]}{\partial \mathbf{y}} - \frac{\partial[\mathbf{d}, [\mathbf{y}, \mathbf{d}]]}{\partial \mathbf{y}}\right)_{\mathbf{y}=\mathbf{0}}\right) \frac{\partial \mathtt{Ad}_{\mathtt{A}}\boldsymbol{\epsilon}}{\partial \boldsymbol{\epsilon}}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}}\\
&= \left(\mathtt{I} + \frac{1}{2}\cdot\frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\Big|_{\mathbf{y}=\mathbf{0}}\right. && \text{(A.46)}\\
&\quad \left. + \frac{1}{12}\left(\frac{\partial[\mathbf{y}, [\mathbf{0}, \mathbf{d}]]}{\partial \mathbf{y}} + \frac{\partial[\mathbf{0}, [\mathbf{y}, \mathbf{d}]]}{\partial \mathbf{y}} - \frac{\partial[\mathbf{d}, \mathbf{w}]}{\partial \mathbf{w}}\Big|_{\mathbf{w}=[\mathbf{y},\mathbf{0}]} \frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\right)\right) \mathtt{Ad}_{\mathtt{A}}\\
&= \left(\mathtt{I} + \frac{1}{2}\cdot\frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}} + \frac{1}{12}\frac{\partial[\mathbf{w}, \mathbf{d}]}{\partial \mathbf{w}}\frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\right) \mathtt{Ad}_{\mathtt{A}} && \text{(A.47)}\\
&= \left(\mathtt{I} + \frac{1}{2}\cdot\frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}} + \frac{1}{12}\left(\frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\right)^2\right) \mathtt{Ad}_{\mathtt{A}}\,. && \text{(A.48)}
\end{aligned}
$$

From line (A.46) to line (A.47), we used that $[\mathbf{d}, \mathbf{w}] = -[\mathbf{w}, \mathbf{d}]$. At $(*)$, the third order Campbell-Baker-Hausdorff expansion is used. If we use the first-order or second-order expansion instead, we receive

$$
\mathtt{J}_{\mathtt{A}}^{\mathtt{B}} := \frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{A}\exp(\boldsymbol{\epsilon})\mathtt{B})^{\vee}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \approx \mathtt{Ad}_{\mathtt{A}}\,, \tag{A.49}
$$

or

$$
\mathtt{J}_{\mathtt{A}}^{\mathtt{B}} := \frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{A}\exp(\boldsymbol{\epsilon})\mathtt{B})^{\vee}\Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \approx \left(\mathtt{I} + \frac{1}{2}\cdot\frac{\partial[\mathbf{y}, \mathbf{d}]}{\partial \mathbf{y}}\right) \mathtt{Ad}_{\mathtt{A}} \tag{A.50}
$$

respectively.

Appendix B

Jacobians

B.1 Projections and Camera Forward Models

Let $\mathbf{a} \in \mathbb{R}^n$. The Jacobian of the projection function,

$$\text{proj} : \mathbb{R}^n \to \mathbb{R}^{n-1}, \quad \text{proj}(\mathbf{a}) = \frac{1}{a_n} \begin{pmatrix} a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} , \tag{B.1}$$

is

$$\frac{\partial \text{proj}(\mathbf{a})}{\partial \mathbf{a}} = \frac{1}{a_n} \begin{bmatrix} \mathtt{I}_{n\times n} & -\frac{1}{a_n} \begin{pmatrix} a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \end{bmatrix} . \tag{B.2}$$

Let $\mathbf{x} \in \mathbb{R}^3$, f being the focal length and $\mathbf{p}$ the principal point. The Jacobian of the monocular forward model,

$$\hat{\mathbf{z}}_{\text{mono}}(\mathbf{x}) = f \cdot \text{proj}(\mathbf{x}) + \mathbf{p} , \tag{B.3}$$

is

$$\frac{\partial \hat{\mathbf{z}}_{\text{mono}}(\mathbf{x})}{\partial \mathbf{x}} = f \cdot \frac{\partial \text{proj}(\mathbf{x})}{\partial \mathbf{x}} = \frac{f}{x_3} \begin{bmatrix} 1 & 0 & -\frac{x_1}{x_3} \\ 0 & 1 & -\frac{x_2}{x_3} \end{bmatrix} . \tag{B.4}$$

Similarly, for the stereo forward model

$$\hat{\mathbf{z}}_{\text{stereo}}(\mathbf{x}) = \begin{pmatrix} f \cdot \text{proj}(\mathbf{x}) + \mathbf{p} \\ f\frac{x_1 - b}{x_3} + p_1 \end{pmatrix} \tag{B.5}$$

we get

$$\frac{\partial \hat{\mathbf{z}}_{\text{stereo}}(\mathbf{x})}{\partial \mathbf{x}} = \frac{f}{x_3} \begin{bmatrix} 1 & 0 & -\frac{x_1}{x_3} \\ 0 & 1 & -\frac{x_2}{x_3} \\ 1 & 0 & -\frac{x_1 - b}{x_3} \end{bmatrix} , \tag{B.6}$$

with b being the baseline.

B.2 Pose-Point Transformations

When a point $\mathbf{y} \in \mathbb{R}^3$ is transformed using the homogeneous (4×4) matrix $\mathtt{T} \in \mathbf{SE}(3)$, we often use the shorthand notation $\mathtt{T} \cdot \mathbf{y}$ for $\mathtt{R}\mathbf{y} + \mathbf{t}$ with $\mathtt{R}$ being the rotation matrix and $\mathbf{t}$ being the translation vector of $\mathtt{T}$. However, in order to derive the Jacobians of such transformations, we need to be more precise. We write instead

$$\text{proj}(\mathtt{T} \cdot \dot{\mathbf{y}}) = \mathtt{R}\mathbf{y} + \mathbf{t} \tag{B.7}$$

with $\dot{\mathbf{y}} = (y_1, y_2, y_3, 1)^\top$ being a function with maps the 3-vector $\mathbf{y}$ to its homogeneous counterpart. Note that the derivative of $\text{proj}(\dot{\mathbf{x}})$ with respect to $\dot{\mathbf{x}}$ is

$$\left.\frac{\partial \text{proj}(\mathbf{q})}{\partial \mathbf{q}}\right|_{\mathbf{q}=\dot{\mathbf{x}}} \overset{(B.2)}{=} \begin{bmatrix} \mathtt{I}_{3\times 3} & -\mathbf{x} \end{bmatrix} , \tag{B.8}$$

since $q_4 = 1$.

In the following, we wish to derive the partial derivatives of the transformation $\text{proj}(\mathtt{T} \cdot \dot{\mathbf{y}})$ with respect to the point $\mathbf{x}$ and pose $\mathtt{T}$. The point Jacobian is simply

$$\frac{\partial \text{proj}(\mathtt{T} \cdot \dot{\mathbf{y}})}{\partial \mathbf{y}} = \frac{\partial(\mathtt{R}\mathbf{y} + \mathbf{t})}{\partial \mathbf{y}} = \mathtt{R} \tag{B.9}$$

As discussed in Section 2.4.10, the Jacobian with respect to a pose $\mathtt{T}$ is calculated using the smooth paths $\mathtt{T}_k(t) := \exp(t\widehat{\mathbf{e}_k})\mathtt{T}$ through $\mathtt{T}$. Thus, the partial derivative of the transformation $\text{proj}(\mathtt{T} \cdot \dot{\mathbf{y}})$ with respect to $\mathtt{T}$ is

$$\begin{aligned} \left.\frac{\partial \text{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \cdot \mathtt{T} \cdot \dot{\mathbf{y}})}{\partial \epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} &= \left.\frac{\partial \text{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \cdot \dot{\mathbf{x}})}{\partial \epsilon_k}\right|_{\substack{\boldsymbol{\epsilon}=\mathbf{0} \\ \mathbf{x}=\mathtt{R}\cdot\mathbf{y}+\mathbf{t}}} \\ &= \left.\frac{\partial \text{proj}(\mathbf{q})}{\partial \mathbf{q}}\right|_{\mathbf{q}=\dot{\mathbf{x}}} \left.\frac{\partial \exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \cdot \dot{\mathbf{x}}}{\partial \epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} \\ &\overset{(2.101)}{=} \begin{bmatrix} \mathtt{I}_{3\times 3} & -\mathbf{x} \end{bmatrix} \cdot \mathtt{G}_k \dot{\mathbf{x}} , \end{aligned} \tag{B.10}$$

with $\mathtt{G}_k$ being the kth generator of $\mathbf{SE}(3)$. For $\mathbf{SE}(3)$ (and $\mathbf{Sim}(3)$ too), the last row of each generator $\mathtt{G}_k$ consists of zeros only (see Appendix A.1). Thus, the last entry of the vector $\mathtt{G}_i\dot{\mathbf{x}}$ is zero too and therefore

$$\left.\frac{\partial \mathrm{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \cdot \dot{\mathbf{x}})}{\partial \epsilon_k}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} = \begin{bmatrix} \mathtt{I}_{3\times 3} & -\mathbf{x} \end{bmatrix} \cdot \mathtt{G}_k\dot{\mathbf{x}} = \begin{bmatrix} \mathtt{I}_{3\times 3} & \mathbf{0} \end{bmatrix} \cdot \mathtt{G}_k\dot{\mathbf{x}} \,, \tag{B.11}$$

Now, we can write down the (3×6) pose Jacobian for $\mathbf{SE}(3)$:

$$\begin{aligned} &\left.\frac{\partial \mathrm{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \cdot \mathtt{T} \cdot \dot{\mathbf{y}})}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} \\ &= \begin{bmatrix} \mathtt{I}_{3\times 3} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \mathtt{G}_1\dot{\mathbf{x}} & \mathtt{G}_2\dot{\mathbf{x}} & \mathtt{G}_3\dot{\mathbf{x}} & \mathtt{G}_4\dot{\mathbf{x}} & \mathtt{G}_5\dot{\mathbf{x}} & \mathtt{G}_6\dot{\mathbf{x}} \end{bmatrix} \\ &= \begin{bmatrix} \mathtt{I}_{3\times 3} & -[\mathbf{x}]_\times \end{bmatrix} \quad \text{with} \quad \mathbf{x} = \mathtt{R} \cdot \mathbf{y} + \mathbf{t} \,. \end{aligned} \tag{B.12}$$

Using similar arguments, we get for $\mathtt{S} \in \mathbf{Sim}(3)$

$$\frac{\partial \mathrm{proj}(\mathtt{S} \cdot \dot{\mathbf{y}})}{\partial \mathbf{y}} = \frac{\partial (s\mathtt{R}\mathbf{y} + \mathbf{t})}{\partial \mathbf{y}} = s\mathtt{R} \tag{B.13}$$

and

$$\left.\frac{\partial \mathrm{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{sim}(3)}) \cdot \mathtt{S} \cdot \dot{\mathbf{y}})}{\partial \boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} = \begin{bmatrix} \mathtt{I}_{3\times 3} & -[\mathbf{x}]_\times & \mathbf{x} \end{bmatrix} \,, \tag{B.14}$$

with $\mathbf{x} = s\mathtt{R} \cdot \mathbf{y} + \mathbf{t}$.

B.3 Inverse Depth Point Transformations

Let $\boldsymbol{\psi} \in \mathbb{R}^3$. The function $\Pi(\boldsymbol{\psi}) = (\frac{\psi_1}{\psi_3}, \frac{\psi_2}{\psi_3}, \frac{1}{\psi_3})^\top$ maps an inverse depth point to its Euclidean counterpart and vice versa. Its Jacobian is

$$\frac{\partial \Pi(\boldsymbol{\psi})}{\partial \boldsymbol{\psi}} = \begin{bmatrix} \frac{1}{\psi_3} & 0 & -\frac{\psi_1}{\psi_3^2} \\ 0 & \frac{1}{\psi_3} & -\frac{\psi_2}{\psi_3^2} \\ 0 & 0 & -\frac{1}{\psi_3^2} \end{bmatrix} = \frac{1}{\psi_3} \begin{bmatrix} 1 & 0 & -y_1 \\ 0 & 1 & -y_2 \\ 0 & 0 & -y_3 \end{bmatrix} \quad \text{with} \quad \mathbf{y} = \Pi(\boldsymbol{\psi}) \,. \tag{B.15}$$

The Jacobian of the transformation $\mathrm{proj}(\mathtt{T} \cdot \dot{\Pi}(\boldsymbol{\psi})) = \mathtt{R} \cdot \Pi(\boldsymbol{\psi}) + \mathbf{t}$ with respect to the inverse depth point $\boldsymbol{\psi}$ is:

$$\begin{aligned} \frac{\partial (\mathtt{R} \cdot \Pi(\boldsymbol{\psi}) + \mathbf{t})}{\partial \boldsymbol{\psi}} &= \frac{\partial (\mathtt{R}\mathbf{y} + \mathbf{t})}{\partial \mathbf{y}} \cdot \frac{\partial \Pi(\boldsymbol{\psi})}{\partial \boldsymbol{\psi}} = \mathtt{R} \cdot \left(\frac{1}{\psi_3} \begin{bmatrix} 1 & 0 & -y_1 \\ 0 & 1 & -y_2 \\ 0 & 0 & -y_3 \end{bmatrix} \right) \\ &= \frac{1}{\psi_3} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & -\mathtt{R}\mathbf{y} \end{bmatrix} \quad \text{with} \quad \mathbf{y} = \Pi(\boldsymbol{\psi}) \,, \end{aligned} \tag{B.16}$$

and $\mathbf{r}_1$, $\mathbf{r}_2$ being the first two column vectors of $\mathtt{R}$.

B.4 Bundle Adjustment

Under the assumption of a monocular camera, the Jacobians of bundle adjustment are

$$\begin{aligned}\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\text{mono}}(\text{proj}(\mathtt{T}\cdot\dot{\mathbf{y}})))}{\partial\mathbf{y}} &= -\left.\frac{\partial\hat{\mathbf{z}}_{\text{mono}}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathtt{R}\mathbf{y}+\mathbf{t}}\cdot\frac{\partial(\mathtt{R}\mathbf{y}+\mathbf{t})}{\partial\mathbf{y}} \\ &= -\frac{f}{x_3}\begin{bmatrix}1 & 0 & -\frac{x_1}{x_3}\\ 0 & 1 & -\frac{x_2}{x_3}\end{bmatrix}\cdot\mathtt{R}\end{aligned} \tag{B.17}$$

and

$$\begin{aligned}&\left.\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\text{mono}}(\text{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\cdot\mathtt{T}\cdot\dot{\mathbf{y}})))}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} \\ &= -\left.\frac{\partial\hat{\mathbf{z}}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathtt{R}\mathbf{y}+\mathbf{t}}\cdot\left.\frac{\partial\text{proj}(\exp(\widehat{\boldsymbol{\epsilon}})\cdot\dot{\mathbf{x}})}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}} \\ &= -\frac{f}{x_3}\begin{bmatrix}1 & 0 & -\frac{x_1}{x_3}\\ 0 & 1 & -\frac{x_2}{x_3}\end{bmatrix}\cdot\begin{bmatrix}\mathtt{I}_{3\times3} & -[\mathbf{x}]_\times\end{bmatrix}.\end{aligned} \tag{B.18}$$

B.5 Anchored Inverse Depth Bundle Adjustment

The reprojection errors of anchored inverse depth bundle adjustment are defined as:

$$\mathbf{z}-\hat{\mathbf{z}}(\text{proj}(\mathtt{T}_{lw}\cdot\mathtt{T}_{aw}^{-1}\cdot\dot{\Pi}(\boldsymbol{\psi}))) \tag{B.19}$$

with $\mathtt{T}_{lw}\in\mathbf{SE}(3)$ being the observer pose, $\mathtt{T}_{aw}\in\mathbf{SE}(3)$ being the anchor pose and $\boldsymbol{\psi}\in\mathbb{R}^3$ the inverse depth point. Furthermore, let $\mathtt{T}_{la}=\mathtt{T}_{lw}\mathtt{T}_{aw}^{-1}$ and $\mathbf{y}=\Pi(\psi)$. Assuming a stereo camera, the inverse depth point Jacobian is:

$$\begin{aligned}&\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\text{stereo}}(\text{proj}(\mathtt{T}_{lw}\cdot\mathtt{T}_{aw}^{-1}\cdot\dot{\Pi}(\boldsymbol{\psi}))))}{\partial\boldsymbol{\psi}} \\ &= -\left.\frac{\partial\hat{\mathbf{z}}_{\text{stereo}}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathtt{R}_{la}\mathbf{y}+\mathbf{t}_{la}}\cdot\frac{\partial(\mathtt{R}_{la}\cdot\Pi(\boldsymbol{\psi})+\mathbf{t}_{la})}{\partial\boldsymbol{\psi}} \\ &= -\frac{f}{\psi_3 x_3}\begin{bmatrix}1 & 0 & -\frac{x_1}{x_3}\\ 0 & 1 & -\frac{x_2}{x_3}\\ 1 & 0 & -\frac{x_1-b}{x_3}\end{bmatrix}\begin{bmatrix}\mathbf{r}_{la1} & \mathbf{r}_{la2} & -\mathtt{R}_{la}\mathbf{y}\end{bmatrix}.\end{aligned} \tag{B.20}$$

with $\mathbf{r}_{la1}$, $\mathbf{r}_{la2}$ being the first two column vectors of $\mathtt{R}_{la}$.

The derivative with respect to the observer pose $\mathtt{T}_{lw}$ is:

$$\begin{aligned}
&\left.\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\text{stereo}}(\text{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\cdot\mathtt{T}_{lw}\cdot\mathtt{T}_{aw}^{-1}\cdot\dot{\Pi}(\boldsymbol{\psi}))))}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}\\
=&\ -\left.\frac{\partial\hat{\mathbf{z}}_{\text{stereo}}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathtt{R}_{la}\mathbf{y}+\mathbf{t}_{la}}\cdot\left.\frac{\partial\text{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\cdot\dot{\mathbf{x}})}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}\\
=&\ -\frac{f}{x_3}\begin{bmatrix}1&0&-\frac{x_1}{x_3}\\0&1&-\frac{x_2}{x_3}\\1&0&-\frac{x_1-b}{x_3}\end{bmatrix}\begin{bmatrix}\mathtt{I}_{3\times3}&-[\mathbf{x}]_\times\end{bmatrix}\ .
\end{aligned}\tag{B.21}$$

Finally, the Jacobian with respect to the anchor pose $\mathtt{T}_{la}$ equals:

$$\begin{aligned}
&\left.\frac{\partial(\mathbf{z}-\hat{\mathbf{z}}_{\text{stereo}}(\text{proj}(\mathtt{T}_{lw}\cdot(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\cdot\mathtt{T}_{aw})^{-1}\cdot\dot{\Pi}(\boldsymbol{\psi}))))}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}\\
\overset{(2.69)}{=}&\ -\left.\frac{\partial\hat{\mathbf{z}}_{\text{stereo}}(\text{proj}(\mathtt{T}_{lw}\cdot\mathtt{T}_{aw}^{-1}\cdot\exp(\widehat{-\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\cdot\dot{\mathbf{y}}))}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}\\
=&\ \left.\frac{\partial(\hat{\mathbf{z}}_{\text{stereo}}(\text{proj}(\mathtt{T}_{la}\cdot\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\cdot\dot{\mathbf{y}}))}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}\\
=&\ \left.\frac{\partial\hat{\mathbf{z}}_{\text{stereo}}(\mathbf{x})}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathtt{R}_{la}\mathbf{y}+\mathbf{t}_{la}}\cdot\frac{\partial\text{proj}(\mathtt{T}_{la}\dot{\mathbf{y}})}{\partial\mathbf{y}}\cdot\left.\frac{\partial\text{proj}(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)})\cdot\dot{\mathbf{y}})}{\partial\boldsymbol{\epsilon}}\right|_{\boldsymbol{\epsilon}=\mathbf{0}}\\
=&\ \frac{f}{x_3}\begin{bmatrix}1&0&-\frac{x_1}{x_3}\\0&1&-\frac{x_2}{x_3}\\1&0&-\frac{x_1-b}{x_3}\end{bmatrix}\cdot\mathtt{R}_{la}\cdot\begin{bmatrix}\mathtt{I}_{3\times3}&-[\mathbf{y}]_\times\end{bmatrix}\ .
\end{aligned}\tag{B.22}$$

B.6 Pose-graph Optimisation

The energy of pose-graph optimisation is the sum of squares over terms of the form

$$\log(\mathtt{T}_{ji}\mathtt{T}_i\mathtt{T}_j^{-1})^\vee\ .\tag{B.23}$$

Here, $\mathtt{T}_{ji}$ is the constant 'measurement' and $\mathtt{T}_i$, $\mathtt{T}_j$ are the variables we wish to modify during the optimisation.

Again, let us first assume $\mathtt{T}_{ji},\mathtt{T}_i,\ \mathtt{T}_j\in\mathbf{SE}(3)$. Then, the residual error is defined as $\begin{pmatrix}\boldsymbol{\tau}\\\boldsymbol{\phi}\end{pmatrix}:=\log(\mathtt{T}_{ji}\mathtt{T}_i\mathtt{T}_j^{-1})^\vee_{\mathfrak{se}(3)}$ with $\boldsymbol{\tau},\boldsymbol{\phi}\in\mathbb{R}^3$. The partial derivative with respect

to $\mathtt{T}_i$ can be calculated as:

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{T}_{ji} \exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \mathtt{T}_i \mathtt{T}_j^{-1})^{\vee}_{\mathfrak{se}(3)} \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \\
&\overset{(A.50)}{\approx} \left(\mathtt{I} + \frac{1}{2} \cdot \frac{\partial \left[\mathbf{y}, \begin{pmatrix} \boldsymbol{\tau} \\ \boldsymbol{\phi} \end{pmatrix} \right]_{\mathfrak{se}(3)}}{\partial \mathbf{y}} \right) \mathtt{Ad}_{\mathtt{T}_{ji}} \\
&\overset{(A.17)}{=} \left(\mathtt{I} + \frac{1}{2} \cdot \begin{bmatrix} -[\boldsymbol{\phi}]_\times & -[\boldsymbol{\tau}]_\times \\ \mathtt{0}_{3\times 3} & -[\boldsymbol{\phi}]_\times \end{bmatrix} \right) \cdot \begin{bmatrix} \mathtt{R}_{ji} & [\mathbf{t}_{ji}]_\times \mathtt{R}_{ji} \\ \mathtt{0} & \mathtt{R}_{ji} \end{bmatrix} .
\end{aligned} \tag{B.24}$$

Here, we approximate the derivative of the matrix logarithm using the second-order Campell-Baker-Haussdorf expansion (see Appendix A.6). Similarly, for the Jacobian with respect to $\mathtt{T}_j$ it holds:

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{T}_{ji} \mathtt{T}_i (\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \mathtt{T}_j)^{-1})^{\vee}_{\mathfrak{se}(3)} \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \\
&\overset{(2.73)}{=} -\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{se}(3)}) \mathtt{T}_j \mathtt{T}_i^{-1} \mathtt{T}_{ji}^{-1})^{\vee}_{\mathfrak{se}(3)} \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \\
&\overset{(A.50)}{\approx} -\left(\mathtt{I} + \frac{1}{2} \cdot \frac{\partial \left[\mathbf{y}, \begin{pmatrix} -\boldsymbol{\tau} \\ -\boldsymbol{\phi} \end{pmatrix} \right]_{\mathfrak{se}(3)}}{\partial \mathbf{y}} \right) \mathtt{Ad}_{\mathtt{I}} \\
&\overset{(A.17)}{=} -\left(\mathtt{I} + \frac{1}{2} \cdot \begin{bmatrix} [\boldsymbol{\phi}]_\times & [\boldsymbol{\tau}]_\times \\ \mathtt{0}_{3\times 3} & [\boldsymbol{\phi}]_\times \end{bmatrix} \right) .
\end{aligned} \tag{B.25}$$

For the group of similarity transformations $\mathbf{Sim}(3)$, the residual error is defined as $(\boldsymbol{\tau}^\top, \boldsymbol{\phi}^\top, \varsigma)^\top := \log(\mathtt{S}_{ji} \mathtt{S}_i \mathtt{S}_j^{-1})^{\vee}_{\mathfrak{sim}(3)}$ with $\varsigma \in \mathbb{R}$ and $\mathtt{S}_{ji}, \mathtt{S}_i, \mathtt{S}_j \in \mathbf{Sim}(3)$. The Jacobians are

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{S}_{ji} \exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{sim}(3)}) \mathtt{S}_i \mathtt{S}_j^{-1})^{\vee}_{\mathfrak{sim}(3)} \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \\
&\overset{(A.50,A.19)}{\approx} \left(\mathtt{I} + \frac{1}{2} \cdot \begin{bmatrix} -[\boldsymbol{\phi}]_\times - \varsigma \mathtt{I} & -[\boldsymbol{\tau}]_\times & \boldsymbol{\tau} \\ \mathtt{0}_{3\times 3} & -[\boldsymbol{\phi}]_\times & \mathbf{0} \\ \mathtt{0}_{1\times 3} & \mathtt{0}_{1\times 3} & 0 \end{bmatrix} \right) \cdot \begin{bmatrix} s_{ji} \mathtt{R}_{ij} & [\mathbf{t}_{ji}]_\times \mathtt{R}_{ji} & -\mathbf{t}_{ji} \\ \mathtt{0}_{3\times 3} & \mathtt{R}_{ji} & \mathbf{0} \\ \mathtt{0}_{1\times 3} & \mathtt{0}_{1\times 3} & 1 \end{bmatrix} ,
\end{aligned} \tag{B.26}$$

and

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\epsilon}} \log(\mathtt{S}_{ji} \mathtt{S}_i (\exp(\widehat{\boldsymbol{\epsilon}}_{\mathfrak{sim}(3)}) \mathtt{S}_j)^{-1})^{\vee}_{\mathfrak{sim}(3)} \Big|_{\boldsymbol{\epsilon}=\mathbf{0}} \\
&\overset{(A.50,A.19)}{\approx} -\left(\mathtt{I} + \frac{1}{2} \cdot \begin{bmatrix} [\boldsymbol{\phi}]_\times + \varsigma \mathtt{I} & [\boldsymbol{\tau}]_\times & -\boldsymbol{\tau} \\ \mathtt{0}_{3\times 3} & [\boldsymbol{\phi}]_\times & \mathbf{0} \\ \mathtt{0}_{1\times 3} & \mathtt{0}_{1\times 3} & 0 \end{bmatrix} \right) .
\end{aligned} \tag{B.27}$$

Bibliography

Adelson, E., Anderson, C., Bergen, J., Burt, P., and Ogden, J. (1984). Pyramid methods in image processing. *RCA engineer*, 29(6):33–41. 72

Agarwal, S., Snavely, N., Simon, I., Seitz, S., and Szeliski, R. (2009). Building Rome in a day. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 21, 82

Agrawal, M. (2006). A Lie algebraic approach for consistent pose registration for general euclidean motion. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 22, 129, 139

Alcantarilla, P., Yebes, J., Almazán, J., and Bergasa, L. (2012). On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 84

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics (T-RO)*, 24(5):1027–1037. 23, 128, 173

Arun, K., Huang, T., and Blostein, S. (1987). Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 9(5):698–700. 129, 159

Bailey, T., Nieto, J., and Nebot, E. (2006). Consistency of the FastSLAM algorithm. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 121

Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework: Part 1. *International Journal of Computer Vision (IJCV)*, 56(3):221–255. 155

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 70, 82, 159

Beall, C., Lawrence, B., Ila, V., and Dellaert, F. (2010). 3D reconstruction of underwater structures. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 4418–4423. 84

Bell, B. and Cathey, F. (1993). The iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297. 96, 120

Besl, P. and McKay, N. (1992). A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256. 175

Betgé-Brezetz, S., Hébert, P., Chatila, R., and Devy, M. (1996). Uncertain map making in natural environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 19, 118

Biber, P. and Duckett, T. (2005). Dynamic maps for long-term operation of mobile service robots. In *Proceedings of Robotics: Science and Systems (RSS)*. 179

Bosse, M., Newman, P., Leonard, J. J., Soika, M., Feiten, W., and Teller, S. (2003). An atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 138

Brown, D. C. (1958). A solution to the general problem of multiple station analytical stereo triangulation. Technical report, Patrick Airforce Base, Florida. 20, 81

Byrod, M. and Astrom, K. (2010). Conjugate gradient bundle adjustment. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 82

Cadena, C., McDonald, J., Leonard, J., and Neira, J. (2011). Place recognition using near and far visual information. In *World Congress of the International Federation of Automatic Control (IFAC)*. 158

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 82

Castellanos, J., Martinez-Cantin, R., Tardós, J., and Neira, J. (2007). Robocentric map joining: Improving the consistency of EKF-SLAM. *Robotics and Autonomous Systems*, 55:21–29. 119

Castellanos, J., Neira, J., and Tardós, J. (2004). Limits to the consistency of EKF-based SLAM. In *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*. 94, 119

Castellanos, J. A. (1998). *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*. PhD thesis, Universidad de Zaragoza, Spain. 19, 118

Castellanos, J. A., Tardós, J. D., and Schmidt, G. (1997). Building a global map of the environment of a mobile robot: The importance of correlations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 18

Castle, R., Klein, G., and Murray, D. (2011). Wide-area augmented reality using camera tracking and mapping in multiple regions. *Computer Vision and Image Understanding (CVIU)*. 173

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applicationsto imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145. 72

Chiuso, A., Favaro, P., Jin, H., and Soatto, S. (2002). Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(4):523–535. 20, 88

Chli, M. and Davison, A. J. (2009). Active Matching for visual tracking. *Robotics and Autonomous Systems*, 57(12):1173 – 1187. Special Issue 'Inside Data Association'. 82, 115

Churchill, W. and Newman, P. (2012). Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 179

Civera, J., Davison, A. J., Magallón, J. A., and Montiel, J. M. M. (2009a). Drift-free real-time sequential mosaicing. *International Journal of Computer Vision (IJCV)*, 81(2):128–137. 116

Civera, J., Davison, A. J., and Montiel, J. M. M. (2008). Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics (T-RO)*, 24(5):932–945. 98

Civera, J., Grasa, O., Davison, A. J., and Montiel, J. M. M. (2010). 1-point RANSAC for EKF filtering. Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5):609–631. 84

Civera, J., Grasa, O. G., Davison, A. J., and Montiel, J. M. M. (2009b). 1-point RANSAC for EKF-based structure from motion. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 88, 119

Clemente, L. A., Davison, A. J., Reid, I., Neira, J., and Tardós, J. D. (2007). Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems (RSS)*. 79, 124, 138

Corke, P., Strelow, D., and Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 84

Cummins, M. and Newman, P. (2008). Accelerated appearance-only SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 173

Cummins, M. and Newman, P. (2009). Highly scalable appearance-only SLAM — FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems (RSS)*. 23, 116, 128, 173

Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*. SIAM. 67, 69, 82, 131

Davison, A. J. (1998). *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford. 18, 19, 118

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 20, 24, 55, 82, 83, 88, 95, 119, 123, 141

Davison, A. J. (2005). Active search for real-time vision. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 24, 70, 82, 115

Davison, A. J., Molton, N. D., Reid, I., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067. 22, 24, 85, 95

Deans, M. and Herbert, M. (2001). Experimental comparison of techniques for localization and mapping using a bearing-only senser. *Experimental Robotics VII*, pages 395–404. 120

Dellaert, F. and Kaess, M. (2006). Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research (IJRR)*, 25:1181–1203. 21, 69, 86, 139

Dudek, G. and Jugessur, D. (2000). Robust place recognition using local appearance based methods. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 173

Dyer, P. and McReynolds, S. (1969). Extension of square-root filtering to include process noise. *Journal of Optimization Theory and Applications*, 3(6):444–458. 96

Eade, E. (2008). *Monocular Simultaneous Localisation and Mapping*. PhD thesis, University of Cambridge. 78, 98, 101, 120, 121

Eade, E. and Drummond, T. (2006). Scalable monocular SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 23, 86, 121

Eade, E. and Drummond, T. (2007). Monocular SLAM as a graph of coalesced observations. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 22, 85, 88, 90, 96, 120, 124, 138, 141

Eade, E. and Drummond, T. (2008). Unified loop closing and recovery for real time monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*. 173

Eade, E. and Drummond, T. (2009). Edge landmarks in monocular SLAM. *Image and Vision Computing (IVC)*, 27(5):588 – 596. 69

Engels, C., Stewénius, H., and Nistér, D. (2006). Bundle adjustment rules. In *Proceedings of Photogrammetric Computer Vision*. 67, 82, 112

Estrada, C., Neira, J., and Tardós, J. D. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics (T-RO)*, 21(4):588–596. 138

Eustice, R. M., Singh, H., and Leonard, J. J. (2005). Exactly sparse delayed state filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 39, 139, 140

Fayad, J., Russell, C., and Agapito, L. (2011). Automated articulated structure and 3d shape recovery from point correspondences. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 179

Finkel, R. A. and Bentley, J. L. (1976). Quad trees a data structure for retrieval on composite keys. *ACTA INFORMATICA*, 4(1):1–9. 74, 83

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. 79, 82

Frese, U. (2006). Treemap: An $O(logn)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122. 140

Gallier, J. (2011). *Geometric Methods and Applications: For Computer Science and Engineering*. Springer, second edition. 47, 52, 53, 186

Galvez-Lopez, D. and Tardós, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics (T-RO)*, 28(5). 172, 174

Geiger, A., Ziegler, J., and Stiller, C. (2011). Stereoscan: Dense 3d reconstruction in real-time. In *Proceedings of the IEEE Intelligent Vehicles Symposium*. 84

Graber, G., Pock, T., and Bischof, H. (2011). Online 3d reconstruction using convex optimization. In *1st Workshop on Live Dense Reconstruction From Moving Cameras @ ICCV*. 180

Grisetti, G., Stachniss, C., Grzonka, S., and Burgard, W. (2007). A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*. 22, 23, 139, 146

Gutmann, J.-S. and Konolige, K. (1999). Incremental mapping of large cyclic environments. In *International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. 139

Gyer, M. S. and Brown, D. C. (1967). The inversion of the normal equations of analytical aerotriangulation by the method of recursive partitioning. Technical report, Rome Air Development Center, Rome, New York. 82

Haralick, R., Lee, C., Ottenberg, K., and Nölle, M. (1994). Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision (IJCV)*. 84

Harris, C. G. and Pike, J. M. (1987). 3D positional integration from image sequences. In *Proceedings of the Alvey Vision Conference*, pages 233–236. 20, 85

Harris, C. G. and Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151. 82, 84

Hartley, R. (1995). In defence of the 8-point algorithm. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 70

Hartley, R. and Schaffalitzky, F. (2004). L-infinity minimization in geometric reconstruction problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 86, 120

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition. 37, 59, 68, 79, 82, 151

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2010). RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*. 175

Highham, N. (1990). Exploiting fast matrix multiplication within the level 3 BLAS. *ACM Transactions on Mathematical Software (TOMS)*, 16. 64

Holmes, S., Sibley, G., Klein, G., and Murray, D. W. (2009). A relative frame representation for fixed-time bundle adjustment in SFM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 119, 173

Horn, B. and Schunck, B. (1981). Determining optical flow. *Artificial Intelligence*, 17:185–203. 71

Howard, A., Sukhatme, G. S., and Mataric, M. J. (2004). Multi-robot mapping using manifold representations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 142, 172

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press. 87

Jeong, Y., Nister, D., Steedly, D., Szeliski, R., and Kweon, I. (2010). Pushing the envelope of modern methods for bundle adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1474–1481. 67, 82, 102, 143, 151

Johannsson, H., Kaess, M., Fallon, M., and Leonard, J. (2012). Temporally scalable visual slam using a reduced pose graph. Technical report, Massachusetts Institute of Technology. 173, 174, 175

Julier, S. and Uhlmann, J. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422. 119

Julier, S. and Uhlmann, K. J. (2001). A counter example to the theory of simultaneous localization and map building. In *IEEE International Conference on Robotics and Automation*. 19, 94, 119

Jung, I. and Lacroix, S. (2003). High resolution terrain mapping using low altitude aerial stereo imagery. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 85, 88, 119, 141

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., and Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research (IJRR)*. To appear. 21, 117, 139

Kaess, M., Ni, K., and Dellaert, F. (2009). Flow separation for fast and robust stereo odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan. 84

Kaess, M., Ranganathan, A., and Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics (T-RO)*, 24(6):1365–1378. 21, 117, 139

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45. 118

Klein, G. and Murray, D. W. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 22, 24, 73, 78, 83, 85, 89, 120, 141, 143, 173

Klein, G. and Murray, D. W. (2009). Parallel tracking and mapping on a camera phone. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 78, 83

Klippenstein, J. and Zhang, H. (2007). Quantitative evaluation of feature extractors for visual SLAM. In *Proceedings of the Canadian Conference on Computer and Robot Vision (CRV)*. 83

Konolige, K. (2010). Sparse sparse bundle adjustment. In *Proceedings of the British Machine Vision Conference (BMVC)*. 67, 69, 82, 143

Konolige, K. and Agrawal, M. (2008). FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics (T-RO)*, 24:1066–1077. 22, 23, 24, 55, 123, 131, 140, 145, 149, 152, 173, 174, 175

Konolige, K., Agrawal, M., and Solà, J. (2007). Large scale visual odometry for rough terrain. In *Proceedings of the International Symposium on Robotics Research (ISRR)*. 24, 84, 141

Konolige, K. and Bowman, J. (2009). Towards lifelong visual maps. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 179

Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B., and Regis, V. (2010). Sparse pose adjustment for 2d mapping. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 69, 139

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011a). g^2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 68, 69, 81, 97, 149, 161

Kümmerle, R., Grisetti, G., and W.Burgard (2011b). Simultaneous calibration, localization, and mapping. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 68

Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., and Kleiner, A. (2009). On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407. 161

Kundu, A., Krishna, K. M., and Jawahar, C. V. (2011). Realtime multibody visual SLAM with a smoothly moving monocular camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 179

Lemaire, T., Lacroix, S., and Solà, J. (2005). A practical 3D bearing-only SLAM algorithm. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 83

Leonard, J. J. and Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the International Workshop on 'Intelligence for Mechanical Systems' @ IROS*. 19, 118

Leutenegger, S., Chli, M., and Siegwart, R. (2011). BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 82

Lim, J., Pollefeys, M., and Frahm, J.-M. (2011). Online environment mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 22, 24, 55, 83, 140, 141, 144, 151, 152, 173, 174

Lourakis, M. I. A. and Argyros, A. A. (2009). SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30. 82

Lovegrove, S. J., Davison, A. J., and Ibanez-Guzmán, J. (2011). Accurate visual odometry from a rear parking camera. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. 68

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 19, 82

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110. 70, 82

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349. 22, 23, 129, 139

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 83

Maddern, W., Milford, M., and Wyeth, G. (2012). CAT-SLAM: probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research*, 31(4):429–451. 174

Mataric, M. (1990). *A distributed model for mobile robot environment-learning and navigation*. PhD thesis, Massachusetts Institute of Technology. 172

McDonald, J., Kaess, M., Cadena, C., Neira, J., and Leonard, J. J. (2011). 6-DOF multi-session visual SLAM using anchor nodes. In *Proceedings of the European Conference on Mobile Robotics (ECMR)*. 179

McLauchlan, P. and Murray, D. (1996). Active camera calibration for a head-eye platform using a variable state-dimension filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 18(1):15–22. 120

McLauchlan, P. F. and Murray, D. W. (1995). A unifying framework for structure and motion recovery from image sequences. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 21, 120

Mei, C., Benhimane, S., Malis, E., and Rives, P. (2008). Efficient homography-based tracking and 3-D reconstruction for single-viewpoint sensors. *IEEE Transactions on Robotics (T-RO)*, 24(6):1352–1364. 130

Mei, C., Sibley, G., Cummins, M., Newman, P., and Reid, I. (2009). A constant time efficient stereo SLAM system. In *Proceedings of the British Machine Vision Conference (BMVC)*. 24, 55, 74, 84, 173, 174

Mei, C., Sibley, G., Cummins, M., Newman, P., and Reid, I. (2010a). RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision (IJCV)*, 94:198–214. 84, 140, 141, 173

Mei, C., Sibley, G., and Newman, P. (2010b). Closing loops without places. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3738–3744. 145, 147, 174

Milford, M., Wyeth, G., and Prasser, D. (2004). Simultaneous localisation and mapping from natural landmarks using ratslam. In *Australasian Conference on Robotics and Automation 2004*. 16, 19, 172

Milford, M. J. and Wyeth, G. (2008). Single camera vision-only SLAM on a suburban road network. In *IEEE International Conference on Robotics and Automation (ICRA)*. 19, 172

Montemerlo, M. and Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 19, 23, 78, 121

Montiel, J. M. M., Civera, J., and Davison, A. J. (2006). Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*. 22, 55, 77, 83, 88, 119

Moravec, H. P. (1980). *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rove*. PhD thesis, Stanford. 19

Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P. (2006). Real-time localization and 3D reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 64, 84, 85, 144, 152

Muja, M. and Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP)*, pages 331–340. 159

Neira, J. and Tardós, J. D. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897. 82, 115

Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 129

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*. 142, 175

Newcombe, R. A., Lovegrove, S., and Davison, A. J. (2011b). DTAM: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 142, 155, 180

Newman, P. (1999). *On the Structure and Solution of the Simultaneous Localization and Map Building Problem*. PhD thesis, University of Sydney. 19, 118

Nistér, D. (2003). Preemptive RANSAC for live structure and motion estimation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 84

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777. 70, 82, 103, 128

Nistér, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 21, 24, 84

Nistér, D., Naroditsky, O., and Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):–. 84, 141

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 23, 116, 128, 173

Nocedal, J. and Wright, S. (2006). *Nummerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, second edition. 31

Olson, E., Leonard, J. J., and Teller, S. (2006). Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 139

Paskin, M. A. (2003). Thin junction tree filters for simultaneous localization and mapping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 19, 138

Paz, L. M., Jensfelt, P., Tardós, J. D., and Neira, J. (2007). EKF SLAM updates in O(n) with divide and conquer SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 138

Pietzsch, T. (2008). Efficient feature parameterisation for visual SLAM using inverse depth bundles. In *Proceedings of the British Machine Vision Conference (BMVC)*. 22, 85, 88, 98, 119

Pinies, P. and Tardós, J. D. (2008). Large scale SLAM building conditionally independent local maps: Application to monocular vision. *IEEE Transactions on Robotics (T-RO)*, 24(5):1094–1106. 124, 138

Pirker, K., Rüther, M., and Bischof, H. (2011). CD SLAM - continuous localization and mapping in a dynamic world. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 85, 173, 174, 179

Ranftl, R., Gehrig, S., Pock, T., and Bischof, H. (2012). Pushing the limits of stereo using variational stereo estimation. In *Proceedings of the IEEE Intelligent Vehicles Symposium)*. 155

Rossmann, W. (2002). *Lie Groups: An Introduction Through Linear Groups*. Oxford Graduate Texts in Mathematics. Oxford University press. 47, 185

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 73, 83

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 82

Rybski, P., Zacharias, F., Lett, J., Masoud, O., Gini, M., and Papanikolopoulos, N. (2003). Using visual features to build topological maps of indoor environments. In *IEEE International Conference on Robotics and Automation*. 83

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260. 84

Scaramuzza, D., Fraundorfer, F., Pollefeys, M., and Siegwart, R. (2009). Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 124

Schönhage, J. and Strassen, V. (1971). Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292. 64

Se, S., Lowe, D., and Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of Robotics Research*, 21(8):735–758. 19

Shewchuk, J. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University. 33, 67

Shi, J. and Tomasi, C. (1994). Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 82

Sibley, G., Matthies, L., and Sukhatme, G. (2005). Bias reduction filter convergence for long range stereo. In *12th International Symposium of Robotics Research*. 38, 96, 120

Sibley, G., Matthies, L., and Sukhatme, G. (2008). A sliding window filter for incremental SLAM. *Unifying perspectives in computational and robot vision*, pages 103–112. 21, 117, 120

Sibley, G., Mei, C., Reid, I., and Newman, P. (2009). Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems (RSS)*. 140, 142, 146, 151, 172

Sim, R., Elinas, P., Griffin, M., and Little, J. J. (2005). Vision-based SLAM using the Rao-Blackwellised particle filter. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*. 23, 86, 121

Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 159, 173

Smith, M., Baldwin, I., Churchill, W., Paul, R., and Newman, P. (2009). The new college vision and laser data set. *International Journal of Robotics Research (IJRR)*, 28(5):595–599. 164

Smith, P., Reid, I., and Davison, A. J. (2006). Real-time single-camera SLAM with straight lines. In *Proceedings of the British Machine Vision Conference (BMVC)*. 69

Smith, R., Self, M., and Cheeseman, P. (1987). A stochastic map for uncertain spatial relationships. In *Workshop on Spatial Reasoning and Multisensor Fusion*. 18, 118

Solà, J., Devy, M., Monin, A., and Lemaire, T. (2005). Undelayed initialization in bearing only SLAM. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 55, 83

Steder, B., Grisetti, G., Grzonka, S., Stachniss, C., Rottmann, A., and Burgard, W. (2007). Learning maps in 3D using attitude and noisy vision sensors. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. 24, 140

Stillwell, J. (2008). *Naive Lie Theory*. Springer. 40, 41, 47, 185

Strasdat, H., Davison, A. J., Montiel, J. M. M., and Konolige, K. (2011). Double window optimisation for constant time visual SLAM. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 169, 175

Strasdat, H., Montiel, J., and Davison, A. (2012). Visual SLAM: Why filter? *Image and Vision Computing (IVC)*. 121

Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010a). Real-time monocular SLAM: Why filter? In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 121

Strasdat, H., Montiel, J. M. M., and Davison, A. J. (2010b). Scale drift-aware large scale monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*. 69, 82, 83, 84, 139, 140, 152

Tardós, J. D., Neira, J., Newman, P., and Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research (IJRR)*, 21(4):311–330. 138

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. Cambridge: MIT Press. 38, 95

Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., and Ng, A. Y. (2002). Simultaneous mapping and localization with sparse extended information filters. In *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics*. 19, 86, 138

Tikhonov, A. and Arsenin, V. (1977). *Solutions of ill-posed problems*. Winston, Washington,DC. 97

Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Technical report, Technical Report CMU-CS-91-132, Carnegie Mellon University. 83

Torr, P. H. S. and Zisserman, A. (2000). MLESAC: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding (CVIU)*, 78(1):138–156. 82

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (1999). Bundle adjustment — a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms, in association with ICCV*. 20, 62, 81, 82, 125, 151

von Neumann, J. (1929). Über die analytischen Eigenschaften von Gruppen lineare Transformationen und ihrer Darstellungen. *Matematische Zeitschrift*, 30:3–42. 40

Walter, M., Eustice, R., and Leonard, J. (2007). Exactly sparse extended information filters for feature-based SLAM. *International Journal of Robotics Research (IJRR)*, 26(4):335–359. 21, 116, 139

Wedel, A., Pock, T., Braun, J., Franke, U., and Cremers, D. (2008). Duality tv-l1 flow with fundamental matrix prior. In *Image and Vision Computing New Zealand (IVCNZ)*. 83

Werlberger, M., Pock, T., and Bischof, H. (2010). Motion estimation with non-local total variation regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 116

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2009). A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*. 174

Williams, V. (2011). Breaking the Coppersmith-Winograd barrier. Technical report, UC Berkeley and Stanford University. 65

Zhang, Z. and Shan, Y. (2003). Incremental motion estimation through modified bundle adjustment. In *Proceedings of the International Conference on Image Processing (ICIP)*. 84