Technische Universität München

Fakultät für Bauingenieur- und Vermessungswesen

WS 2012/13

Robust and Efficient Monocular SLAM

Master's Thesis by Matthias Meßner

Supervisor: Prof. Dr. Richard Bamler
Advisors: Dr.-Ing. Markus Ulrich
Dr.-Ing. Stefan Auer
Submission date: 30. April, 2013

Technische Universität München
Fakultät für Bauingenieur- und Vermessungswesen
Masterarbeit
WS 2012/13

Robust and Efficient Monocular SLAM

Themensteller:

Prof. Dr. Richard Bamler[1]
Masterarbeit
Fakultät für Bauingenieur- und Vermessungswesen
Technische Universität München
Arcisstraße 11, 80333 München

Bearbeitet von:

Matthias Meßner
Gerhart-Hauptmann-Ring 10, 81737 München
Matrikelnummer: 3034519 , Fachsemester: 11
Betreuer: Dr.-Ing. Stefan Auer[1]
Dr.-Ing. Markus Ulrich[2]

Abgabetermin: 30. April 2013

---

[1] Fakultät für Bauingenieur- und Vermessungswesen, Technische Universität München, Arcisstr. 11, 80333 München
[2] MVTec Software GmbH, Neherstr. 1, 81675 München

Selbstständigkeitserklärung

Erklärung gemäß §18 Absatz 9 APSO der Technischen Universität München:

„Ich versichere, dass ich diese Master's Thesis selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe."

Matthias Meßner
München, den 30.04.2013

Abstract

This Master's thesis deals with the issue of the three dimensional reconstruction of an environment that has been recorded with a camera, while simultaneously determining the location of the said camera relative to the created map. This problem is called Simultaneous Localization and Mapping (SLAM). To solve this task, two different methods are presented in this thesis.

Both are based on an approach that was developed in the area of Augmented Reality (AR). While the first technique is almost identical to the AR approach, the second one has been subject to some modifications. These substitute the detection of salient image points for another algorithm. Additionally, the detected points are represented by clearly distinguishable descriptors, allowing for the location in other pictures more easily.

Both approaches have been evaluated with various datasets, and subsequently analysed in regards to accuracy, robustness and scale-drift. Furthermore, the suitability of both methods for the recreation of the environment was examined.

The conducted experiments show that the modified technique exhibits remarkably higher accuracy and diminished scale-drift. The robustness of the procedure is improved as well, as the descriptor can still be found even after large variations in appearance. The unmodified method on the other hand is better suited for mapping the recorded surroundings, as these are described with more points.

Kurzfassung

Diese Masterarbeit beschäftigt sich mit dem Problem, die mit einer Kamera aufgenommene Umgebung dreidimensional zu rekonstruieren, während gleichzeitig die Position der Kamera relativ zu dieser erstellen Karte bestimmt wird. Dieses Problem nennt sich gleichzeitige Lokalisierung und Kartenerstellung (Simultaneous Localization and Mapping (SLAM)). Um diese Aufgabe zu lösen, werden in dieser Arbeit zwei verschiedene Verfahren vorgestellt.

Beide basieren auf einem Ansatz, der im Bereich der Augmented Reality (AR) entwickelt wurde. Während das erste Verfahren beinahe identisch mit diesem Ansatz ist, wurden im zweiten Modifikationen vorgenommen. Diese ersetzen die Detektion der markanten Punkte durch einen anderen Algorithmus. Zusätzlich werden die detektierten Punkte durch stark unterscheidbare Deskriptoren repräsentiert, was ein leichteres Auffinden in anderen Bildern ermöglicht.

Beide Ansätze wurden anhand unterschiedlicher Datensätze evaluiert und auf Genauigkeit, Robustheit und Skalendrift hin analysiert. Zusätzlich wurde untersucht, inwieweit die beiden Verfahren zur Rekonstruktion der Umgebung geeignet sind.

Die durchgeführten Experimente zeigen, dass das modifizierte Verfahren deutlich bessere Genauigkeiten und verminderten Skalendrift aufweist. Zusätzlich wird die Robustheit des Verfahrens erhöht, da die Deskriptoren auch bei großen Veränderungen wiedergefunden werden können. Im Gegensatz dazu ist das unveränderte Verfahren besser geeignet, die aufgenommene Umgebung zu rekonstruieren, da diese mit mehr Punkten beschrieben wird.

Acknowledgements

First and foremost, I want to express my sincere gratitude to my supervisors Dr.-Ing. Stefan Auer and Dr.-Ing. Markus Ulrich for continuously discussing problems and supporting me throughout my entire work. They sacrificed a lot of time to give me helpful advice and useful suggestions.

Furthermore, I am very thankful to my unofficial second supervisors Dr.-techn. Friedrich Fraundorfer and Bertram Drost. Both assisted me and shared their enormous knowledge with me.

I also want to thank my lab partners. Especially Christina Eisenhofer and Tobias Böttger who always had an open ear for my questions and explained mathematical problems every time I required it. Additionally, they found time to proofread my thesis while working on their own at the same time.

I would like to thank Roman Moie for his programming skills and Andreas Pumpf as well as Tobias Koch for having a look at my work.

Also some special thanks to my friends Fabian Lehn and Thomas Wanninger who proofread my thesis and gave me a lot of hints concerning the English language.

Last but not least, I would like to thank my family. They gave me the possibility to study, showed a lot of understanding and encouraged me to complete my work on the highest possible standard.

This work was conducted in collaboration with the Technical University of Munich and the MVTec Software GmbH. I appreciated the great support I was given and the provision of the workspace.

Contents

1 Introduction **1**
1.1 Thesis Outline 4

2 Related Work **5**
2.1 Probabilistic Approaches 7
  2.1.1 Extended Kalman Filter 7
  2.1.2 Particle Filter 9
2.2 Geometric Approaches 10
2.3 Comparison 13

3 Theory and Background **15**
3.1 Rigid Transformations 15
3.2 Perspective Camera Model 17
3.3 Correspondence Problem 20
  3.3.1 Feature Extraction 21
  3.3.2 Feature Matching 22
3.4 Two-View Geometry 23
  3.4.1 Camera Geometry (Motion) 27
  3.4.2 Scene Geometry (Structure) 28
3.5 Multiview Reconstruction 30
  3.5.1 Bundle Adjustment 30
  3.5.2 Robust Bundle Adjustment 36

4 Proposed Method **41**

5 Localization **45**
5.1 Local Localization 46
  5.1.1 Prediction 47
  5.1.2 Pose Refinement 48
5.2 Discussion 50

6 Map Building **53**
6.1 Map Architecture 54
6.2 Map Initialization 55

6.3 Map Extension 56
6.4 Discussion 58

7 Experiments 61
7.1 Datasets 61
    7.1.1 New College Dataset 62
    7.1.2 Middlebury Dataset 64
    7.1.3 Office Scene 64
7.2 Evaluation 65
    7.2.1 Evaluation Using Patches 65
    7.2.2 Evaluation Using SIFT-Features 74
7.3 Comparison 78
7.4 Processing time 80
7.5 Discussion 81

8 Conclusion and Future work 83

CHAPTER 1

Introduction

A few decades ago, autonomous robots supporting humans with their daily tasks were only pipe dreams of visionaries and elements of science fiction literature and movies. The times of R2-D2 and C-3P0 from Star Wars or the positronic robots from Asimov's robot series have not dawned yet, but the development in the field of autonomous robots and self-driving vehicles has progressed remarkably in the past twenty-five years.
Encouraged by competitions such as RoboCub which has been arranged since 1997, already today robots with a variety of different applications exist. A well-known and impressive example for specialised operations is the exploration of Mars by the robotic rover Curiosity [25]. But there are also various demonstrations in the service domain, such as tour guide robots [12] or lawn mowers [38]. Probably the most successful one is iRobots' vacuum cleaner Roomba with over six million units sold world wide at the beginning of 2011 [1] .

(a) Google's driverless car [14] (b) Mars rover Curiosity [25] (c) Service Robot PR2 [60]

Figure 1.1: Three different examples of currently existing autonomous robots and self-driving cars.

In the field of self driving cars great strides have been made as well. This is mainly due to the DARPA Grand Challenge [56], in which an unmanned vehicle has to cover a certain distance through a desert at a given time. Whereas in the first challenge in 2004 not a single car reached the end of the course, one year later already five vehicles crossed the finish line. Furthermore it came to the first overtaking manoeuvre between two autonomous vehicles in history, when the car from Stanford university pulled ahead of Carnegie Mellon's H1ghlander. Under the supervision of Sebastian Thrun, leader of DARPA's winning team in 2005, the Google driverless cars were developed driving now accident-free for over 500000 km. Moreover as of September 2012 four U.S. States legalized driverless cars and a new DARPA challenge for robots has been announced as well.

Of course the development is still in its infancy but these examples show that the dream of fully autonomous robots which take over life-endangering tasks such as rescuing victims can come true one day.

In order to accomplish these tasks there are mainly three things that all robots and autonomous cars need: A map, their position within this map and sensors to perceive their environment.

If a robot has an accurate map it could deduce its position within the map using its sensors. Unfortunately in many applications such as the mars exploration a map cannot be provided. Sometimes it may exist in terms of a computer-aided design model (CAD model) which was blueprinted by an architect. As a matter of fact these models are often not up to date and even worse not compatible with the robot's sensor. But even if a sufficient map was available it would assume a static environment. This does not represent the real world since living creatures and objects change their position with varying speed. In order to operate in these highly dynamic environments and make autonomous driving or human robot interaction possible, the robot has to create a map from scratch. Unfortunately, building an up to date map requires the knowledge of the robot's position. Sometimes this can be retrieved from absolute measurements like a Global Navigation Satellite System (GNSS) but most of the time the robot is either not equipped with an appropriate sensor or no signal is available like in indoor applications or planet explorations.

Thus, the robot is situated in some kind of a chicken-egg problem: An accurate map is necessary to deduce its location whereas a map can only be built if a precise pose is known. The only solution is to estimate the map and the position simultaneously. Or in other words: The robot has to answer two fundamental questions at the same time: “Where am I?” and “What is the structure of my surroundings?”. In literature this problem is called Simultaneous Localization and Mapping (SLAM) and a solution to it gives the answer to both questions.

Unfortunately the robot's sensors have a limited accuracy and neither the true map nor the true position can be determined. Therefore a SLAM algorithm has to calculate the “best” map and the “best” robot positions given the noisy sensor measurements.

As a consequence the noise of the used sensor plays a key role in the accuracy of SLAM and a variety have been used for SLAM. Famous examples of sensors mainly used at the beginning of SLAM solutions are laser range finders [15] and sonars [59]. On the one hand these sensors have the advantage that the robot is directly informed about the three-dimensional structure of the observed scene. On the other hand these sensors are heavy, bulky, expensive and resource-hungry.

Another commonly employed sensor is a video camera. Contrary to other sensors a camera is cheap, small and lightweight. Furthermore cameras provide a vast amount of data at a high sampling rate. Together with the improvements made in computer vision this rich data can not only be used to accomplish relatively simple tasks such as localization but also more complex ones such as object detection and even scene understanding. This is clearly not possible from simple geometric sensors such as a laser range finder.

A downside of cameras is that they do not provide direct information about the depth of the observed scene. Instead the 3D information has to be reconstructed using at least two images of the same scene taken from different points of view. Furthermore the reconstruction is only possible if the observed scene has sufficient texture.

Despite these drawbacks cameras are often the sensor of choice in SLAM yielding the field of visual SLAM. However, this is not really surprising since humans perceive up to 80% of their environment with their eyes and have been successfully exercising SLAM over a long time now. Humans persistently see, feel, hear, taste and smell their surroundings and combine these senses to a three dimensional colored map of the environment with sounds and odours. Together with this map their brain delivers the position within this map. With this knowledge humans can continuously describe their surrounding and even interact with the environment in order to accomplish tasks such as planning the fastest path from A to B or dodging obstacles. All things considered this is relatively easy for humans and they hardly have to think about it. But in fact visual perception in particular is not completely understood. Although remarkable progress has been made in recent years as shown above, it remains to be seen if mankind can artificially recreate perception or even surpass itself.

Building on recently presented methods, this thesis intents to estimate the camera pose in a previously unknown environment while building an accurate map of the observed structure. Although it would be much easier, the application is not restricted to robots equipped with one or more cameras. Instead, robust tracking should be possible if the camera undergoes rapid motions and accelerations as they may occur in the hand-held case. Furthermore, it is presumed that only a monocular camera is used that operates in a static environment. However, in respect of an extension to highly dynamic environments methods that additionally take these case into consideration would be useful and are therefore preferred.

The focus of mapping lies on the accurate reconstruction of an environment of limited size. A sparse representation is sufficient for the desired tasks especially because a subsequent multi-view reconstruction based on a variational approach is intended.

Moreover, the thesis concentrates on an efficient solution to both, tracking and mapping.

1.1 Thesis Outline

Chapter 2: In this chapter an overview of the development of SLAM is given. Further some solutions to the SLAM problem together with seminal works are presented. Each solution is discussed in detail and finally the choice of one method is justified due to a comparison between the previously described algorithms.
Chapter 3: This chapter describes the mathematical background needed for this thesis. Starting with a general camera model the correspondence problem and two-view geometry are covered as well. Finally multi-view geometry is explained with a huge part about the theory of bundle adjustment.
Chapter 4: This chapters presents a general overview of the proposed method.
Chapter 5: This chapter will answer one fundamental question of SLAM: “Where am I?”
Chapter 6: In this chapter the answer to the second fundamental question of SLAM is given: “What is the structure of my surrounding?”
Chapter 7: Together with an overview of the dataset the proposed method is evaluated in this chapter.
Chapter 8: This chapter concludes this thesis and provides an outlook for future work.

CHAPTER 2

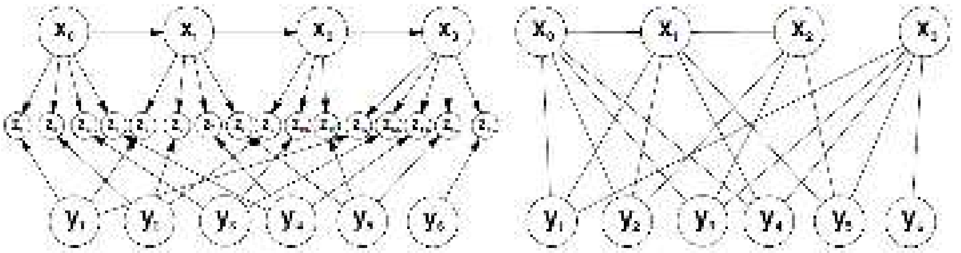
Related Work

Originally, SLAM has been intensively researched in the field of mobile robotics, with the goal that a robot can autonomously navigate in a previously unknown area while simultaneously creating a map from scratch. Since a robot needs its position and the map immediately, because important tasks such as obstacle avoidance and human-robot interactions depend upon them, researchers focused on real-time solutions. As time passed, many SLAM systems with a plethora of different sensor types were presented.

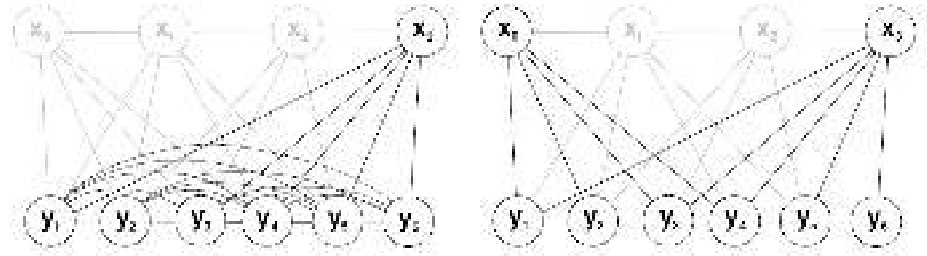
Recently, researchers became aware that SLAM with cameras as primary sensor has already been tackled in the field of computer vision where it is called structure from motion (SfM). SfM can be seen as the problem of simultaneously estimating the camera poses and the scene structure given (unordered) images. Hence, this is essentially the same problem as SLAM only using a camera as a sensor. However, the main focus of SfM is the accurate reconstruction of the observed scene and not processing time. Furthermore, the whole data is usually accessible and not acquired over time.

Because of the fundamentally divergent claims and purposive ideas, a connection between SLAM and SfM could not been established at first and it took years to understand that both speak a uniform language. Recently, after gaining a better understanding of the localization and reconstruction process the gap between the two fields of research could have finally been bridged.

In order to relate to this process, the SLAM problem is expressed with regard to different graphical models such as a Bayesian network as shown in figure 2.1(a). In this network, each variable x_i represents a (historic) six degree of freedom camera pose and each variable y_j the three dimensional coordinates of a feature in the map. If a feature y_j has been measured from a pose x_i, both are linked by a feature measurement z_k. For example, from position x_0 only the features y_1 to y_4 were measured by the four measurements z_1 to z_4.



(a) Bayesian Network (b) Markov Random Field



(c) Filter (d) Keyframe BA

Figure 2.1: (a) represents the SLAM problem as a Bayesian Network. The goal is to determine the poses x_i and the map features y_j given noisy measurements z_k (b) Markov random field representation for the SLAM problem. The measurements are not shown explicitly. (c) Inference graph in filtering methods. After every step all past poses are marginalised out and only the current one is maintained. (d) Inference graph for keyframe bundle adjustment. Only a small subset of all poses and their corresponding measurements are kept. The other poses together with their measurements are discarded. [51]

Every time a new image is acquired, a new camera pose is added and new links are established by measuring existing features y_j. From time to time, new features have to be added because the camera is exploring unmapped areas. As a consequence, the network will grow arbitrarily large over time.

Another way to express the SLAM problem is by a Markov random field in which the measurements z_k are not shown explicitly but represented by links between poses and features. This is shown in figure 2.1(b)

It is well known, that bundle adjustment (BA) is the optimal solution to the SLAM problem as it provides the maximum likelihood estimate to the graph of figure 2.1(b) assuming Gaussian noise [51]. Since the graph continuously grows over time, the computational complexity of BA, which is cubic in the number of camera poses, quickly prevents a real-time solution to the problem. Therefore, other methods have been developed in order to provide an accurate solution while simultaneously reducing the computational complexity.

One of the first solutions to SLAM was presented by Moutarlier and Chatila [30] in 1990 using an extended Kalman filter (EKF). This and other filtering methods were very popular between the early nineties and about the following 15 years. The basic procedure of these methods is illustrated in figure 2.1(c).

Only the current pose and all features in the map are maintained in a joint probability distribution which is propagated through time. All other poses are marginalised out after each step. This results in a compact, but fully interconnected graph that does not grow arbitrarily with time. However, updating the joint posterior becomes very costly once the number of features is large. Hence, approaches based on filtering methods are usually restricted to maps of small size. The other possibility is to continue using bundle adjustment but to thin out the graph as it is shown in figure 2.1(d). Only a small subset (x_0 and x_3) of all poses together with their measurements are retained, whereas the other poses (x_1 and x_2) and their corresponding measurements are not considered in the optimization process. This yields a sparsely interconnected graph with more components compared to filtering approaches. The advantage of this is that the optimization of the graph is sustainable even if the map consists of a large number of features.

2.1 Probabilistic Approaches

Solving the SLAM problem requires the concurrent determination of a map m along with the robot's pose x_t at time t given noisy sensor measurements $z_{1:t}$ and control inputs $u_{1:t}$. From a probabilistic perspective this can be formulated as follows:

$$p(x_t, m | z_{1:t}, u_{1:t}),$$

where p denotes the joint posterior probability distribution over the current pose along with the map.

A recursive formula for computing the joint posterior at time t is given by the Bayes filter [55]. Unfortunately, this estimator requires the solution of an integral which cannot be calculated in closed-form. Therefore, the Bayes filter as such is not directly applicable to SLAM and instead, the posterior has to be approximated. For this, various parametric and nonparametric implementations exist. Two important representatives, the extended Kalman filter (EKF) [55] and the particle filter [55], are briefly described in the following.

2.1.1 Extended Kalman Filter

The usage of the extended Kalman filter (EKF) as a solution to the SLAM problem was first introduced through Smith et al. [46] in the late 1980s. The EKF approximates the SLAM posterior with a multi-variate Gaussian distribution [55].

After the first implementation due to Moutarlier and Chatila [30] in 1990 the EKF was successfully employed with various sensors such as sonars [54] or laser range finders [5] in both, static [55] and dynamic environments [58].

A huge number of researchers [33], [32] and [21] also adopted the EKF to the field of visual SLAM. One of the most pioneering works in this domain was published in 2003 by Davison [10]. His approach was able to estimate the pose of a hand-held camera in real-time while concurrently creating a sparse map of the observed scene. However, this implementation and all EKF-based approaches in particular have some drawbacks that will be pointed out in the following.

Inconsistencies

It is shown in a series of works [20], [8], [7] and [2] that the EKF yields inconsistent results.

Julier and Uhlman [20] for instance state in a clear mathematical proof that the EKF is doomed to produce an inconsistent map. They confirm their analytical results in a simple scenario with a stationary vehicle observing one beacon. Although theoretically not possible, the orientation covariance begins to decline and quickly drops below its initial value. This feeds back into the EKF yielding an inconsistent landmark and vehicle estimate. They further extend their researches to a moving vehicle showing that the EKF is not able to produce long-term consistent maps.

In simulated experiments Castellanos et al. [8] corroborate the results of [20]. Linearization errors are determined to be responsible for the map becoming inconsistent. As an improvement, the paper suggests a new mapping technique that reduces the linearization errors by keeping the map estimate in the reference frame of the vehicle yielding improved map consistency. Despite these efforts it is likely that the filter becomes inconsistent again and they suggest to investigate alternative solutions to the SLAM problem. The benefits of using a robot centered representation are extended in their latter work [7].

Bailey et al. [2] show in a series of experiments in a two dimensional environment that the principle cause of the inconsistencies is the vehicle heading uncertainty. As soon as this uncertainty exceeds a limit, the feature estimates show excessive information gain and the EKF becomes inconsistent. This can only be avoided, if the Jacobians are linearized about the unavailable ground-truth state. However, the authors notice that inconsistencies can be handled if the heading uncertainty is bound by periodically taken direct observations of the orientation.

Computational Effort

Another key limitation of EKF-based approaches lies in the computational complexity. Even if only a single feature is observed, every element in the covariance matrix maintained by the EKF has to be updated with a total cost of $\mathcal{O}(N^2)$ for N features in the map. This quadratic complexity limits the map size of EKF-based approaches to a few hundreds. However, due to the requirement of maintaining a scarce map a new problem arises which is often denoted as the *fundamental dilemma of EKF-SLAM* [55]: In EKF-SLAM the standard approach to data association is the usage of a maximum likelihood (ML) estimator [55].

Although this might be suitable for dense maps it does not work well for sparse maps. Since the EKF-SLAM maintains a joint posterior over the current pose along with the map, incorporating wrong measurements can never be undone and can yield an irretrievable corruption of the map. Hence, researches had to make huge efforts to prevent this from happening. Davison [11] for example used active search in combination with Joint Compatibility Branch and Bound (JCCB) [34] to reject spurious matchings.

2.1.2 Particle Filter

Particle filter approximate the Bayesian posterior probability distribution by a set of random samples with associated weights [55]. Unfortunately, particle filters scale exponentially with the dimension of the state space and are thus not applicable to the SLAM problem. However, Montemerlo et al. [29] showed that the posterior can be factored into a posterior over the robot's trajectory and a product of N individual feature posteriors that are only conditioned on the trajectory estimate. The posterior over the robot's path is estimated by a particle filter consisting of M particles, whereas each feature posterior is approximated by independent EKFs. Combining a particle filter and Kalman filters is commonly known as a Rao-Blackwellized particle filter.

The resulting SLAM algorithm is called FastSLAM and requires an update time of $\mathcal{O}(MN)$ for M particles and N features in the map. It can also be implemented efficiently by using a tree-based data structure which reduces the update time to $\mathcal{O}(M \log N)$.

The approach was evaluated in several simulated and real-world experiments. There it was shown, that FastSLAM is able to successfully maintain 50000 features using only 100 particles. FastSLAM exhibits some advantages in terms of data association and complexity [2]. However, there are also a few downsides. Bailey et al. [2] analysed the approach and stated that the algorithm cannot produce long-term consistent estimates of the posterior probability distribution. This behaviour is inevitable and prevents FastSLAM from being an acceptable Bayesian estimator.

Nevertheless, FastSLAM was also adopted to the field of visual SLAM by Sim et al. [44]. They equipped a robot with a stereo rig and attempted to build a metric map of the observed environment. Thereto, they extracted SIFT-features [27] and triangulated three dimensional map points by using the acquired stereo images. In contrast to other SLAM approaches the robot's odometry is not examined further, but instead the motion estimate between two consecutive frames is based on visual odometry.

They evaluated their approach on a 67.5m trajectory collecting 4000 images. Although they were able to obtain a map with more than 11000 three dimensional map points, the average processing time for each frame was 11.9 seconds and thus far from real-time.

2.2 Geometric Approaches

As mentioned at the beginning of this chapter, the optimal solution to the SLAM problem is obtained through bundle adjustment (BA) since it provides the maximum likelihood estimate to the graph depicted in figure 2.1(a) under the assumption of Gaussian noise [17]. Unfortunately, the graph grows arbitrarily in time and the computational complexity of BA soon prevents a real-time solution. To overcome this problem, only some well-chosen poses and their corresponding measurements are retained effectively thinning out the graph.

This concept was first introduced in 2007 by Klein and Murray [22] in their pioneering algorithm Parallel Tracking and Mapping (PTAM). The fundamental idea of their approach was to split tracking and mapping into two different subtasks each running on a separate thread. On the one hand there is a tracking thread, that is responsible to estimate the pose of a single camera. Since many decisions like rendering AR graphics depend upon it this thread has to run in real-time. On the other hand there is a mapping thread, which creates an accurate three dimensional map of the environment. Instead of updating the map at every frame, only some well chosen frames, the so-called keyframes are used for map extension.

At first, the map is constructed using a standard stereo initialization approach. Whenever a new frame is acquired, hundreds of map points are projected into the current frame and identified using a patch-based approach. All recognized map points are then used to refine the camera pose by iteratively minimizing a robust objective function of the reprojection error. As soon as the camera leaves the mapped region the map is extended by adding a new keyframe and new points. Since the extension of the map does not have to be done at every frame, the approach employs the computationally demanding but highly accurate bundle adjustment for structure and motion refinement.

Although PTAM achieved superior tracking performance in comparison with former state-of the art approaches the proposed method suffered a few drawbacks that led to a series of improvements.

Especially during rapid camera motions tracking often fails because motion blur slices out corners of the image. Klein and Murray [23] extended their previous work by adding edgelets to the map. Thereby, they gain resilience to motion blur which made it possible to track the camera even under fast motions. Furthermore, they added a simple relocalisation mechanism that helps the system to recover from tracking failures.

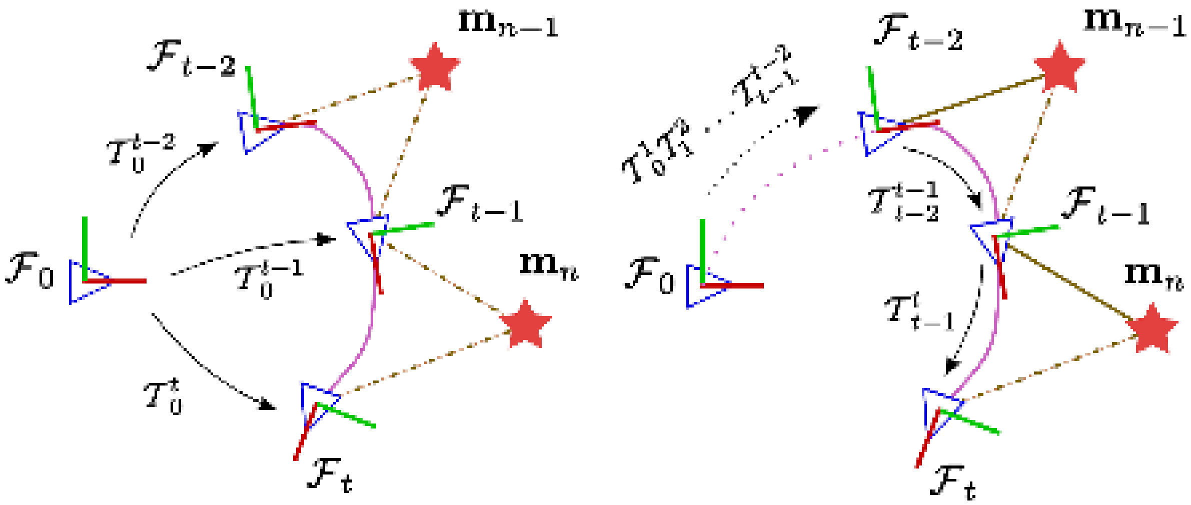
They also adapted their first approach, PTAM, on an iPhone [24] showing that real-time visual SLAM is also possible with commodity hardware. However, they had to lower one's sights in terms of accuracy and robustness.

Although PTAM was able to build an accurate map of the observed scene, the approach was limited to small workspaces. This is mainly due to the complexity of BA which is cubic in the number of keyframes. As soon as the map has reached a certain size the processing time prevents real-time exploration.

One approach to overcome these limitations is to decompose the global map into many local maps of bounded complexity. Hence, PTAM was extended by Castle et al. [9] to support multiple maps. The resulting work is called Parallel Tracking and Multiple Mapping (PTAMM). Instead of waiting the BA to converge they simply construct a new map which allows a much better exploration. However, a fusion of these submaps is never intended.

Another development ascribed to the computational costs of BA is the relative bundle adjustment (RBA). Instead of representing all variables with respect to a single world coordinate frame, Sibley et al. [43] proposed a relative formulation (see figure 2.2). This prevents the propagation of large errors around the loop and even loops with large displacements can be closed in constant time.

However, the relative representation has two main disadvantages. The first one is, that a single global Euclidean representation is never computed. If the global position is required the relative representations have to be transformed into a single Euclidean coordinate system which is computationally demanding and has to be outsourced. The second problem is the decreasing accuracy of RBA if many loops have to be closed.



(a) Fixed coordinate system

(b) Relative representation

Figure 2.2: Absolute and Relative Representation. (a) All map points and camera poses are represented with respect to a single coordinate system. (b) A camera is represented with respect to its previous pose. 3D points are stored relative to their poses. [28]

Despite these drawbacks the relative bundle adjustment is successfully used in many accurate SLAM approaches such as [18], [28] or [50]. Holmes et al. [18] combine the RBA with PTAM in a monocular approach. They show that by temporarily removing measurements the RBA can be partitioned into a local and a global subtask. The local version only optimizes a fixed number of poses around the current one and runs in constant time, whereas the global version performs standard BA in the background. However, the approach lacks convincing experiments and the discussion of large loop closures.

Mei et al. [28] used RBA and a stereo camera for their highly accurate relative SLAM (RSLAM). They evaluated their approach on several large-scale indoor and outdoor datasets. Even in the largest sequence of 2.26 km they achieved an accuracy of a few meters. They further combined their approach with FABMAP for loop detection improving the accuracy after loop closure by a factor of 100. Furthermore, they compared their framework with PTAM in three different sequences gaining superior results in the outdoor as well as in the indoor experiment. Only in a third small desktop environment their performance lagged behind PTAM.

The latter approach of Strasdat et al. [50] combined the RBA with a pose-graph optimization introducing the double window framework. Due to this framework the algorithm achieves not only high local accuracy but can also handle rapid explorations and large loop closures in real-time. They further show in simulated and real experiments with a monocular camera that the system is also able to correct scale-drift in constant time.

All methods described above only construct a sparse representation of the environment. These sparse maps are useful to keep track of a camera in real-time but they do not allow an interaction between robots or virtually placed characters and the observed scene. With the increasing computational power a new field of visual SLAM opened up by representing the world through a dense model.

Probably the first who combined SLAM with the reconstruction of a dense map in real-time were Newcombe and Davison [35] and almost simultaneously Stühmer et al. [49]. Both algorithms are very similar in terms that they both use the PTAM framework for SLAM however, the procedure for obtaining the dense map is different. The approach of [35] first estimates a dense depth map of neighbouring frames using an optical flow algorithm and subsequently fuse these maps to obtain a global dense model, whereas the latter introduced a novel variational approach that directly computes a depth field from multiple images. Although both algorithm constructed a dense map in real-time they did not exploit the map further and tracking still relied on the feature-based system PTAM.

Recently, Newcombe et al. [36] presented a novel approach named Dense Tracking and Mapping (DTAM). The algorithm is leaned against PTAM in terms that tracking and mapping is split and that only a small subset of past poses and images are stored as keyframes. However, each keyframe is additionally assigned a dense depth map which is constructed similar to [49]. In contrast to the latter the algorithm is able to immediately make use of the reconstructed dense 3D surface model for accurate camera tracking.

In comparison with PTAM, the proposed algorithm outperforms the feature-based approach in terms of accuracy and robustness especially under rapid camera motions. However, the system still has to be initialized by a feature-based approach and it is further not able to close large loops.

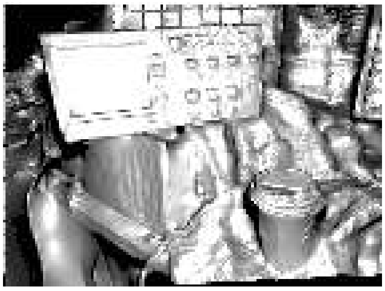


Figure 2.3: Inverse depth map as reconstructed by DTAM [36].

2.3 Comparison

In this chapter two fundamentally different approaches for solving the SLAM problem have been outlined. One solution is based on filtering methods that were very popular at the beginnings of SLAM in the early nineties and approximately until 2007. Filtering methods propagate a joint posterior of the current pose along with the map through time. Unfortunately, the suggested filtering methods cannot produce consistent estimates.

The other solution to SLAM is based on an optimization approach and became very popular in recent years. The key idea is, to maintain only a subset of historic poses and keep all the measurements connected to them.

However, it remains unclear, whether the propagation of a joint probability distribution over all features along with the current pose yields a better accuracy, or if maintaining a small subset of historic poses together with their measurements is more suitable for solving the SLAM problem. The answer to this was first given by Strasdat et al. [51] in 2010 and recently extended in 2012 [53]. They compared two state-of-the-art monocular SLAM systems in simulated experiments as well as in real image sequences with respect to accuracy and computational cost. They stated that systems based on bundle adjustment are superior in terms of accuracy. Only if low processing-power is available, filter approaches are beneficial.

Up to now, approaches based on particle filters were leaved out of this discussion but the same authors argue that it is not necessary to represent approximately Gaussian distributions by non Gaussian ones.

Confirmed by the results of Strasdat et al. [51] the usage of an optimization approach seems more appropriate than the filtering methods. Especially the approach of Klein and Murray [22] is promising to meet the requirements for the goals strove in this thesis. Therefore, the methods presented in this master thesis are based on their seminal work. An overview of the approach can be found in chapter 4, whereas differences between PTAM and the proposed method are explained in chapter 5 and chapter 6 in greater detail.

CHAPTER 3

Theory and Background

Throughout this thesis vectors are written in lower case letters printed in bold type: For example $\boldsymbol{t}$, $\boldsymbol{\omega}$.

An n-dimensional point is either given in Euclidean space $\mathbb{R}^n$ or as a homogeneous vector in Euclidean space $\mathbb{R}^{n+1}$. For example a three dimensional point $\boldsymbol{X}$ is represented as $\boldsymbol{X} = (X, Y, Z)^T$ or as a homogeneous 4-vector $\boldsymbol{X} = (X, Y, Z, 1)^T$ where the last coordinate is conveniently normalized to 1.

Furthermore, points are usually expressed with respect to a particular coordinate system which is signed by a subscript. For example a point $\boldsymbol{X}_w$ may be specified in the world coordinate system w.

3.1 Rigid Transformations

Sometimes it is necessary to consider a point in a different coordinate system. Therefore, it has to be transformed into the other system by a rigid transformation as shown in figure 3.1. A rigid transformation preserves distances and angles between points and may be written as

$$\mathbf{X}_c = R\mathbf{X}_w + \boldsymbol{t}, \tag{3.1}$$

where $\boldsymbol{t} = (t_x, t_y, t_z)^T$ is a translation vector and R is a 3x3 rotation matrix, with the properties that $det(R) = 1$, $R^T = R^{-1}$ and $R^T R = I$.

Using homogeneous coordinates equation (3.1) can be written as

$$\mathbf{X}_c = \begin{pmatrix} R & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}. \tag{3.2}$$

In both cases a point $\boldsymbol{X}_w$ in the world coordinate system is transformed into a point $\boldsymbol{X}_c$ in a coordinate system c.

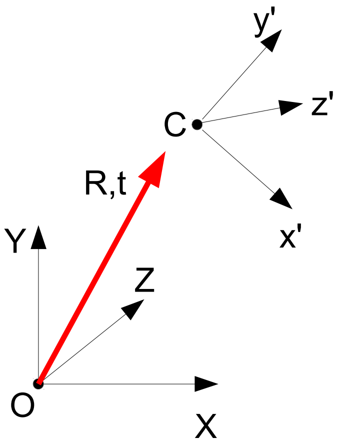


Figure 3.1: A rigid transformation between two coordinate systems. Both systems are related through a rotation matrix R and a translation vector $\boldsymbol{t}$.

The rotation matrix R contains nine elements while only three parameters are sufficient to describe a three dimensional rotation [17]. Hence, R is over parameterized and in respect of large optimization problems a minimal representation is desired.
There are several ways to represent rotations but in the following and throughout this thesis the angle-axis representation as illustrated in figure 3.2 is used.

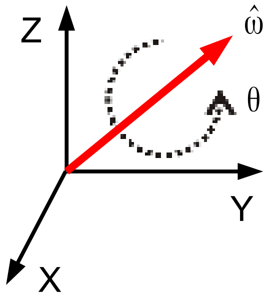


Figure 3.2: Angle-axis representation. A rotation matrix can be parameterized through a unit vector $\hat{\boldsymbol{\omega}}$ in direction of the rotation axis and an angle of rotation θ.

Each rotation can be parameterized by a unit vector $\hat{\boldsymbol{\omega}}$ in the direction of the rotation axis and the angle of rotation $\theta = \|\boldsymbol{\omega}\|$ with $\boldsymbol{\omega} = \theta\hat{\boldsymbol{\omega}}$.

The rotation matrix R corresponding to an angle-axis representation is given by the Rodrigues formula [31]:

$$R = e^{[\hat{\boldsymbol{\omega}}]_\times \theta} = I + \sin\theta [\hat{\boldsymbol{\omega}}]_\times + (1 - \cos\theta)[\hat{\boldsymbol{\omega}}]_\times^2, \tag{3.3}$$

where I denotes the identity matrix and $[\hat{\boldsymbol{\omega}}]_\times$ defines the 3x3 skew-symmetric matrix for $\hat{\boldsymbol{\omega}} \in \mathbb{R}^3$

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

Turning back to the rigid transformation in equation (3.2) a minimal parameterization is given by a 6-vector $(\boldsymbol{\omega}, \boldsymbol{u})^T$ where $\boldsymbol{u} = (u_1, u_2, u_3)$ is a rotated version of the translation $\boldsymbol{t}$ and $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$ is the angle-axis representation of the rotation R. Then the exponential map can be used to transform this 6-vector into the representation of equation (3.2):

$$\exp(\omega, u) = \begin{pmatrix} \exp(\omega) & Vu^T \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} R & \boldsymbol{t} \\ \mathbf{0}^T & 1 \end{pmatrix}, \tag{3.4}$$

where $\exp(\omega)$ is given by the Rodrigues formula (3.3), $V = I + \frac{1-\cos(\theta)}{\theta^2}\theta_\times + \frac{\theta - \sin(\theta)}{\theta^3}\theta_\times^2$ and $\theta = \|\boldsymbol{\omega}\|$.

3.2 Perspective Camera Model

Consider a 3D point $\mathbf{X}$ in a world coordinate system, then a camera describes the mapping between $\mathbf{X}$ and its projection onto the two-dimensional image plane. In order to characterize this mapping, a specific model for the camera in use is necessary. Many different models have been developed over time but the focus of this thesis is on the pinhole camera which performs a perspective projection. The whole mapping process is displayed in figure 3.3.

The 3D point $\mathbf{X}_w = (X_w, Y_w, Z_w)^T$ in the world coordinate system (WCS) is mapped to the two-dimensional point $\mathbf{x}$, where a line joining the world point and the camera centre C intersects the image plane (dotted line). The image plane itself is situated at a distance f behind the camera centre.

In order to project the point $\mathbf{X}_w$ into the image, it first has to be transformed from the WCS into the camera coordinate system (CCS). The CCS is a three-dimensional coordinate system with origin in the camera centre. The relation between the WCS and the CCS is a rigid transformation as illustrated in figure 3.1. Hence, the point $\mathbf{X}_w$ in the WCS can be transformed to the point $\mathbf{X}_c$ in the CCS as explained in section 3.1.

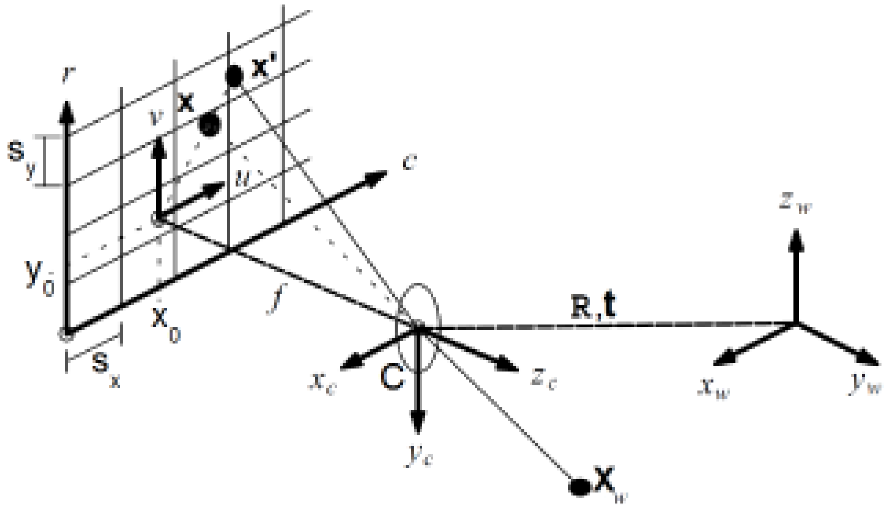


Figure 3.3: Camera model for a pinhole camera. A point $\boldsymbol{X}_w$ in the world coordinate system is first transformed into the camera coordinate system. Subsequently, the point is projected into the image plane coordinate system (u, v) and finally transformed into the image coordinate system (c, r). Image adapted from [47]

The next step is the transition from Euclidean 3-space $\mathbb{R}^3$ to Euclidean 2-space $\mathbb{R}^2$ by projecting the point $\mathbf{X}_c$ into the image plane coordinate system (IPCS). By similar triangles the coordinates of the resulting point $\mathbf{x} = (u, v)^T$ are given by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{Z_c} \begin{pmatrix} X_c \\ Y_c \end{pmatrix}. \tag{3.5}$$

Using homogeneous coordinates this projection can be rewritten as

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX_c \\ fY_c \\ Z_c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix}, \tag{3.6}$$

where $\mathbf{x} = (fX_c, fY_c, Z_c)^T$ is a homogeneous 3 x 1 vector representing the image point and $\mathbf{X}_c = (X_c, Y_c, Z_c, 1)^T$ is a homogeneous 4 x 1 vector representing the same point in the CCS. As one can see, the nonlinear mapping from equation (3.5) can now be simply expressed as a linear mapping between homogeneous points.

Next, the point $\mathbf{x}$ has to be transformed from the image plane coordinate system into the image coordinate system (ICS). As shown in figure 3.4 the origin of the ICS is in the lower left corner and not in the principal point $\mathbf{p}$ as it was assumed in equation (3.6). Additionally, there is a scale difference between the IPCS and the ICS. Denote by s_x and s_y the pitch between the sensor elements in world units in the c and r direction respectively, then equation (3.6) alters to

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} \alpha_x X + Z x_0 \\ \alpha_y Y + Z y_0 \\ Z \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix}, \tag{3.7}$$

where $\alpha_x = \frac{f}{s_x}$ and $\alpha_y = \frac{f}{s_y}$ is the focal length represented in pixel units in the c and r direction and $\tilde{\boldsymbol{x}}_0 = (x_0, y_0)$ are the pixel coordinates of the principal point.

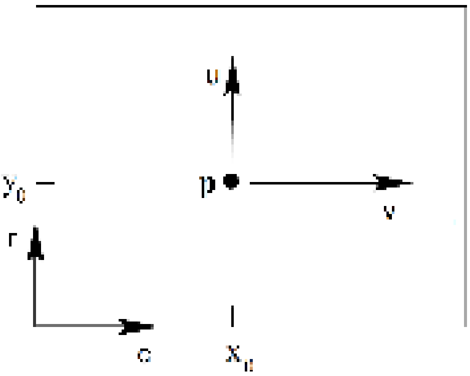


Figure 3.4: Image plane coordinate system (IPCS) and image coordinate system (ICS). Modified from [17]

Extracting

$$K = \begin{pmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

from equation (3.7) it can be rewritten in a more compact form given by

$$\mathbf{x} = K[I|\mathbf{0}]\mathbf{X}_c. \tag{3.9}$$

In this equation $[I|\mathbf{0}]$ is a matrix, where the first part I corresponds to the 3x3 identity matrix and the second part to a column vector with zeros as entries. The 3x3 matrix K is called *camera calibration matrix* and its parameters describe the characteristics of the camera. Since K contains all the information of the camera's internal configuration the parameters are often denoted internal camera parameters or the internal orientation of the camera.

Combining equation (3.2) with equation (3.9) the projection of a point $\mathbf{X}_w$ in the WCS to a point $\mathbf{x}$ in the ICS is given by

$$\mathbf{x} = K[R|\boldsymbol{t}]\mathbf{X}_w, \tag{3.10}$$

where a more compact form is given by

$$\mathbf{x} = P\mathbf{X}_w. \tag{3.11}$$

The resulting 3x4 matrix $P = K[R|\boldsymbol{t}]$ is called *camera projection matrix* and describes the central projection of a pinhole camera.

Since the calibration matrix K has 4 degrees of freedom (DOF) and the exterior orientation has 6 DOF (3 from the rotation and 3 from the translation),q the resulting projection matrix P has 10 degrees of freedom. In some cases an additional parameter, the skew parameter s, is considered increasing the DOF of P from 10 to 11.

The camera model described above is only valid for an optimal pinhole camera. Due to imperfections in the production of the lens the projected point does not lie on a straight line joining $\mathbf{X}_w$ and the camera centre C. Instead, lens distortions alter the location of point $\mathbf{x}$ to a point $\mathbf{x}'$ as it is illustrated in figure 3.3.

According to [47] most of the lens distortions ocurring in the machine vision domain can be modeled adequately by a radial distortion model. Employing this model the undistorted image coordinates $(\tilde{u}, \tilde{v})^T$ are given by

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \frac{2}{1 + \sqrt{1 - 4\kappa(u^2 + v^2)}} \begin{pmatrix} u \\ v \end{pmatrix},$$

where κ describes the magnitude of the radial distortion.

Besides the radial distortion a vast amount of other models exist and the interested reader is invited to consult [47] for more informations.

Usually the four elements in K $(\alpha_x, \alpha_y, x_0, y_0)$ and the distortion coefficient κ are determined in a calibration process as described for example in [61].

It should be noted that throughout this thesis the knowledge of the intrinsic parameters is assumed and that any distortions are removed beforehand.

3.3 Correspondence Problem

In the last chapter the mapping of a three dimensional scene onto a two-dimensional image plane was discussed. Since one main part of SLAM is the creation of a three dimensional map, the inverse problem, reconstructing the scene structure, has to be handled as well. Unfortunately, this is not possible from one view and at least two images of the same scene taken from different points of view are necessary.

Each point in 3D space projects onto a point in an image plane. By finding image points in at least two views that correspond to the same 3D point, the reconstruction of that specific point is possible and will be described in chapter 3.4.2.

In literature finding these pairs of matching points is known as the correspondence problem and can be considered as one of the hardest but most important problems in structure from motion and thus in visual SLAM.

For example, only a small region of both images may overlap or objects seen in one view may be occluded in the second view. Furthermore, positional noise and different conditions during the image acquisition make the search very difficult. To overcome these problems, there exists a vast amount of different approaches.

One basic idea is to represent the image with a sparse but salient set of points usually denoted points of interest or keypoints. In order to guarantee that the same keypoint can be found in a second image despite arbitrary changes of appearance the point has to exhibit some properties:

- stable and reliable re-detection despite arbitrary motions and variations in appearance such as illumination changes
- accurate re-detection in the presence of noise
- distinctiveness from its immediate vicinities

Popular keypoints used in computer vision are Harris corners [16], FAST corners [39] or Difference of Gaussian (DoG) keypoints [27].

3.3.1 Feature Extraction

A keypoint itself is not very well discriminable and in order to identify it in another view, it has to be equipped with additional information. For that, two different approaches are presented in the following.

Patch-based approach

Around a $n \times n$ neighbourhood of the detected keypoint the raw gray values of the image are extracted yielding a so-called patch or template. This patch represents a small, local region of the whole image and acts as a descriptor of the keypoint. Finding corresponding points is usually performed by comparing this descriptor with a small subset of patches extracted around the keypoints in a second image. Between the emerged pairs the similarity is measured and the more similar two patches are, the more probably it is that the represented points describe the same point in 3-space. The whole matching process will be described in chapter 3.3.2 in greater detail.

Feature-based approach

Instead of using the raw gray values, feature-based descriptors characterize the points of interest by extracting information based on local image properties.

A very popular feature-based approach is the Scale Invariant Feature Transform (SIFT) developed by David G. Lowe [26] in 1999. Although there exist many other promising methods such as Speeded Up Robust Features (SURF) [4], a short description of SIFT is given in the following since it will be used later on.

The first step of the algorithm is the detection and localisation of potential keypoints. This is done by extracting the extrema in a previously calculated Difference of Gaussian (DoG) scale-space. Extrema are found by comparing each pixel in this set of DoG images with its 26 neighbours in 3x3 regions of scale-space. If the examined pixel is a local minimum or maximum in its neighbourhood, it is designated as a candidate for a keypoint. The exact location of the candidate point is then found by fitting a 3D quadratic function around its coarse position.

Once possible candidates are found, they are examined further and discarded if they do not meet certain requirements.

Subsequently, an orientation and a scale is assigned to each remaining keypoint based on the gradient magnitude and orientation in a certain region around the point.

In order to characterize the appearance of each keypoint, a SIFT descriptor (SIFT-D) is finally computed. The result is a 128 byte feature vector which is invariant to image scale and rotation and robust against image noise, illumination and viewpoint changes.

A general drawback of feature-based approaches is the high computational costs for keypoint detection and descriptor extraction.

3.3.2 Feature Matching

In order to find corresponding keypoints, the procedure for both, the patch-based and the feature-based approach is essentially the same. Each descriptor of the first image is compared with descriptors extracted around all keypoints in another view. Sometimes additional knowledge is available which helps to restrict the matching to a small area within the second image. Subsequently, the similarity between each emerging pair is measured and if certain requirements are fulfilled, a keypoint is considered as found. However, the similarity measures between both methods differ and will be discussed in the following.

Patch-based matching

In order to obtain a similarity measure between two patches, the patch-based approach uses the raw grey values and calculates some sort of pixel wise correlation. There exists a vast amount of different ways to measure the similarity and two of the simplest ones are the Sum of Absolute Differences (SAD) and the sum of squared differences (SSD) [47].

The main advantage of these measures is that their calculation can be done efficiently. A downside of both is that they are not robust against illumination changes.

Therefore, a similarity measure that is invariant against linear lighting changes is desired.

This can be achieved by the normalized cross correlation (NCC) [47]. However, in contrast to the similarity measures dicussed above, the calculation is more complex and the NCC cannot handle nonlinear illumination changes.
In general, finding correspondences due to a patch-based approach has the advantage of low computational costs. However, this is coupled to some downsides. All similarity measures discussed above are prone to illumination changes. Furthermore, they cannot cope with huge rotations and arbitrary viewpoint and scale changes.

Feature-based matching
In feature-based approaches a keypoint is usually matched to another by finding its nearest neighbour among all detected keypoints in a second view. Typically, the nearest neighbour is the keypoint who's descriptor exhibits the minimum Euclidean distance to the descriptor of the query keypoint. Despite the distinctiveness of the SIFT features mismatches may occur and an additional criterion is needed to discard spurious matchings. Lowe [27] proposed a method that calculates the distance ratio between the closest and the second-closest neighbour. If this ratio is greater than a certain threshold the match is rejected.
A main drawback of feature-based approaches in general and SIFT in particular is the huge computational costs. All stages, keypoint detection, feature extraction and matching usually prevent a real-time implementation and researchers focused on reducing these costs [4].

3.4 Two-View Geometry

One fundamental task of SLAM is the three dimensional reconstruction of the observed environment. If only a monocular camera is used, building a map from scratch is not possible from one view. Instead, two images of the same scene taken from different points of view have to be considered. The underlying projective geometry between two views is the epipolar geometry which is only dependent on the cameras' internal orientations and relative pose [17]. Once the relation between the two-views is known, the 3D structure can be reconstructed.

Let $\boldsymbol{x}$ be the projection of the 3D point $\boldsymbol{X}$ in one view and $\boldsymbol{x}'$ the projection of the same point in another view. The two camera centers are denoted by C and C', respectively. As shown in figure 3.5a the 3D point $\boldsymbol{X}$, the image points $\boldsymbol{x}$ and $\boldsymbol{x'}$ and the camera centers lie in a common plane, the epipolar plane π. The line joining the two camera centers C and C' is known as baseline. The points where the baseline and the image plane intersect are the epipoles denoted with e and e' (see figure 3.5b). The intersection of an epipolar plane with an image plane corresponds to a line which is known as the epipolar line l and l', respectively.
Assuming that only the image point $\boldsymbol{x}$ in one view is known, then the position of the corresponding point $\boldsymbol{x'}$ is constrained according to the epipolar geometry. As it can be seen from figure 3.5b the back-projection of the image point $\boldsymbol{x}$ is a ray in 3D-space defined by $\boldsymbol{x}$ and the camera center C.

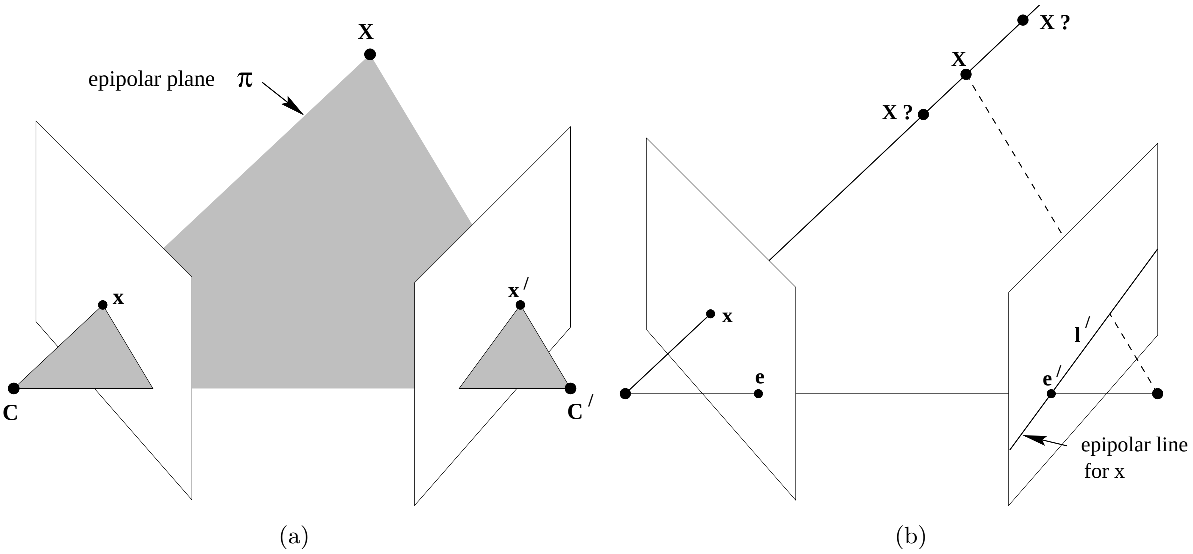


Figure 3.5: Scheme of the epipolar geometry [17]. (a) Both camera centres C and C', the 3D point $\boldsymbol{X}$ and its corresponding image points $\boldsymbol{x}$ and $\boldsymbol{x'}$ lie in the epipolar plane π. (b) The ray back-projected from the image point $\boldsymbol{x}$ is imaged as the epipolar line l' in the second view.

The epipolar plane π is spanned by this ray and the baseline and as shown above, the image point $\boldsymbol{x'}$ also lies in this plane. Since the epipolar plane intersects the second image plane at the epipolar line l', the unknown point $\boldsymbol{x'}$ is forced to lie on this line.

As a consequence, instead of searching the point $\boldsymbol{x'}$ corresponding to $\boldsymbol{x}$ over the entire image plane it can be restricted to just the epipolar line l'.

In mathematical terms the epipolar geometry can be formulated by the fundamental matrix F. As shown in the figure 3.5 for each point $\boldsymbol{x}$ in the first image the matching point $\boldsymbol{x'}$ in the second view lies on the corresponding epipolar line l'. This can be formulated as a projective mapping from $\boldsymbol{x}$ to its corresponding epipolar line l':

$$x \mapsto l'$$

This mapping is represented by the fundamental matrix F and one may write

$$l' = F\boldsymbol{x}$$

A geometric and algebraic derivation of F is treated in detail in [17].

The fundamental matrix F is a homogeneous 3x3 matrix of rank 2. Although it has nine entries there is an overall scale ambiguity and F additionally satisfies the constraint $det\ F = 0$ leaving seven degrees of freedom in total.

Furthermore, the fundamental matrix fulfils an additional property which can easily be derived from the following equations. Given two matching points $\boldsymbol{x}$ and $\boldsymbol{x}'$, then $\boldsymbol{x}'$ lies on the epipolar line l':

$$\boldsymbol{x}'^{T} l' = 0 \tag{3.12}$$

Using

$$l' = F\boldsymbol{x} \tag{3.13}$$

yields the epipolar constraint:

$$\boldsymbol{x}' F \boldsymbol{x} = 0 \tag{3.14}$$

A consequence of equation 3.14 is that the fundamental matrix F can be calculated from image correspondences alone. This will be described in the following.

For homogenous points $\boldsymbol{x} = (x, y, 1)$ and $\boldsymbol{x}' = (x', y', 1)$ a linear equation of the form

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

is obtained.

If n point correspondences are found, a set of n linear equations can be arranged:

$$\underbrace{\begin{pmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{pmatrix}}_{A} \boldsymbol{f} = 0,$$

where $\boldsymbol{f} = \begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{pmatrix}^T$.

Since the vector $\boldsymbol{f}$ can only be determined up to a scale, at least eight point matches are needed for a linear solution. Usually more than eight correspondences are available and due to noise $A\boldsymbol{f} = 0$ cannot be satisfied exactly. Therefore, a vector $\boldsymbol{f}$ has to be found that minimizes $\|A\boldsymbol{f}\|$ subject to $\|\boldsymbol{f}\| = 1$. This can be achieved by employing the singular value decomposition (SVD) as described in [17]. The solution for $\boldsymbol{f}$ in a least-squares sense is then given by the smallest singular value of A.

Due to noise in the measured image coordinates the determined fundamental matrix F usually is not of rank 2 but rather of rank 3. However, the rank 2 constraint can be enforced by the employing the SVD: Let $F = Udiag(d_1, d_2, d_3)$ be the SVD of the fundamental matrix F. Then a matrix F' of rank 2 has to be found that minimizes the Frobenius norm $\|F - F'\|$ subject to $det(F') = 0$. The closest rank 2 approximation of F is then obtained by setting the smallest singular value to zero:

$$F' = Udiag(d_1, d_2, 0)V^T$$

The algorithm described above is known as the 8-point algorithm. However, the results are severely dependent on the choice of the coordinate system. A normalization of the image coordinates before calculating the fundamental matrix yields improved accuracy [17].

Although the normalized 8-point algorithm usually provides satisfying results, it is not the optimal way of determining the fundamental matrix. An overview of methods that either minimize an algebraic or a geometric error are given in [17].

Throughout this thesis it is assumed that the calibration matrix K is known a priori. This additional knowledge can be used to compute the so-called essential matrix E instead of the fundamental matrix F.

Similar to the fundamental matrix, the essential matrix is a homogeneous 3x3 matrix that satisfies the epipolar constraint

$$\hat{\boldsymbol{x}}'^T E \hat{\boldsymbol{x}} = 0,$$

where $\hat{\boldsymbol{x}} = K_1^{-1}\boldsymbol{x}$ and $\hat{\boldsymbol{x}}' = K_2^{-1}\boldsymbol{x'}$ are the normalized image coordinates for matching points $\boldsymbol{x} \leftrightarrow \boldsymbol{x'}$. The matrices K_1 and K_2 are the calibration matrices of the first and second view. Usually it can be assumed that $K_1 = K_2$.

The essential matrix has to fulfil two additional properties as stated in [17]:

A 3x3 matrix is an essential matrix if and only if two of its singular values are equal, and the third is zero.

Furthermore it can be shown that the essential matrix encapsulates the rotation R and the translation $\boldsymbol{t}$ between two views.

$$E = [t]_\times R, \tag{3.15}$$

where $[t]_\times$ is the skew-symmetric matrix defined in chapter 3.1.

The essential matrix has five degrees of freedom (DOF): The rotation matrix R has three DOF and the translation $\boldsymbol{t}$ additively three. However, one DOF has to be subtracted since the essential matrix is only determined up to an overall scale.

The essential matrix can either be computed from the fundamental matrix $E = K_2^T F K_1$ or by the normalized 8-point algorithm. However, since E has only five degrees of freedom it is also possible to calculate it from at least five point correspondences. This is shown in [37] and [48].

Regardless of whether the fundamental matrix F or the essential matrix E has been computed, it was assumed that the given point correspondences match perfectly. In real world applications this cannot be guaranteed because both, the patch-based and the feature-based matching will produce outliers. In order to obtain a correct solution it is necessary to discard these spurious matchings. One possibility is to compute a robust estimate for either E or F employing RANSAC [13] as it is for example described in [17].

3.4.1 Camera Geometry (Motion)

Given a fundamental matrix F or an essential matrix E it is possible to extract the camera matrices P and P' of the two views. Without loss of generality the first camera matrix is assumed to be $P = [I|\mathbf{0}]$. Then, in case of the fundamental matrix, a corresponding camera matrix of the second view is obtained by

$$P' = [[e']_\times F|e'],$$

where e' is the epipol of the second view. A full derivation can be found in [17]
Note that this is only an arbitrary solution since from a given fundamental matrix the pair of camera matrices can only be determined up to a projective transformation. All possible solutions are given by

$$P' = [[e']_\times F + e'v^T|\lambda e'],$$

where λ is a scalar ($\lambda \neq 0$) and v is an arbitrary 3-vector.

The extraction of the camera matrices from the essential matrix is a bit more lavishly. Adopting the following theorem from [17] and [37] the rotation R and the translation $\boldsymbol{t}$ can be recovered:

Assume that the singular value decomposition (SVD) of an essential matrix E is given by $Udiag(1,1,0)V^T$, where U and V are chosen such that $det(U) > 0$ and $det(V) > 0$. Then $\boldsymbol{t} = U(0,0,1)^T = \boldsymbol{u_3}$ and R is either $R_a = UDV^T$ or $R_b = UD^TV^T$, with $D = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Furthermore it is assumed that the first camera matrix $P = [I|\mathbf{0}]$ and that $\|\boldsymbol{t}\| = 1$.
Since the essential matrix is defined up to a scale, the sign of $\boldsymbol{t}$ remains unknown. Combined with the two rotation matrices R_a and R_b this yields four possible choices for the camera matrix P':

$$P'_A = [R_a| + \boldsymbol{u_3}]$$

or

$$P'_B = [R_a| - \boldsymbol{u_3}]$$

or

$$P'_C = [R_b| + \boldsymbol{u_3}]$$

or

$$P'_D = [R_b| - \boldsymbol{u_3}].$$

However, only one of the four possibilities gives the correct solution. Another one corresponds to a reversed direction of the translation vector. The remaining two solutions are obtained by rotating the cameras P'_A and P'_B 180 degrees about the baseline.
An illustration of the four possible solutions is shown in figure 3.6. Only in the top left picture a reconstructed scene point would lie in front of both cameras. Hence, in order to obtain the correct solution for the camera matrix P', it is sufficient to calculate the depth of a single point as it is explained in [17].
It should be noted, that the possibility to restrict the camera matrix P' to only one solution does not solve the scale ambiguity and thus, the camera matrices retrieved from the essential matrix are only determined up to a scale.

3.4.2 Scene Geometry (Structure)

As shown in chapter 3.4.1 a pair of camera matrices P and P' can be extracted from F or E given point correspondences $\boldsymbol{x}$ and $\boldsymbol{x}'$ only. For the following it is assumed that these correspondences are provided and that both camera matrices are known with high accuracy.
If there were no errors in the measured points $\boldsymbol{x}$ and $\boldsymbol{x}'$ a 3D point $\boldsymbol{X}$ in space could simply be reconstructed by back projecting the rays from $\boldsymbol{x}$ and $\boldsymbol{x}'$. Due to noise in the measured points the rays will not intersect in 3-space and the equations

$$\boldsymbol{x} = P\boldsymbol{X} \quad \boldsymbol{x}' = P'\boldsymbol{X}$$

cannot be satisfied exactly for any $\boldsymbol{X}$.

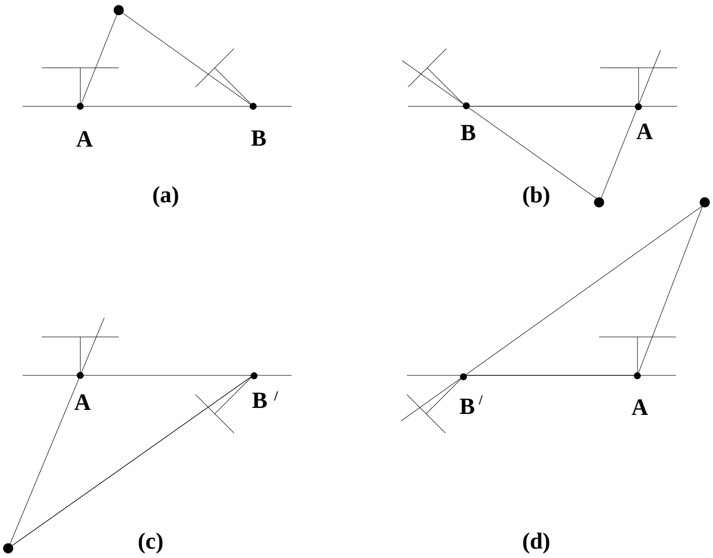


Figure 3.6: From an essential matrix E four possible solutions for the second camera matrix P' are obtained. The difference between the left and the right sides is a reversal of the direction of the translation. The top and bottom rows differ by a rotation through 180° of the camera B about the baseline. Only with the camera matrices extracted from the solution illustrated in (a) the reconstructed points lies in front of both cameras.

There exist various methods to reconstruct a point, but in the following the linear triangulation algorithm as for example explained in [17] is presented:
At first, the unknown scale factor is eliminated by forming a cross product $\boldsymbol{x} \times (P\boldsymbol{X}) = 0$. This gives rise to three equations of the following form:

$$x(p^{3T}X) - (p^{1T}X) = 0$$

$$y(p^{3T}X) - (p^{2T}X) = 0$$

$$x(p^{2T}X) - y(p^{1T}X) = 0,$$

where p^{iT} denotes the i-th row of P. Since only two of these equations are linearly independent, the first two are considered for the reconstruction process. After analogously setting up the equations for the second view, a linear equation system of the form $A\boldsymbol{X} = 0$ with

$$A = \begin{pmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{pmatrix}$$

is composed. A solution for $\boldsymbol{X}$ that minimizes $\|A\boldsymbol{X}\|$ subject to the condition $\|\boldsymbol{X}\| = 1$ can be obtained by using the singular value decomposition (SVD) as discussed in [17].

The linear triangulation method usually yields acceptable results but the estimate of the reconstructed 3D points is not optimal. Assuming a Gaussian error distribution a Maximum Likelihood Estimate (MLE) can be obtained by non-iterative minimizing of a geometric error cost function as it is shown in [17].

The accuracy of a reconstructed point in space depends on the noise in the measured image coordinates and the angle between the back-projected rays as illustrated in figure 3.7. Due to the small parallax, forward motion in particular yields a higher uncertainty in the depth estimates.

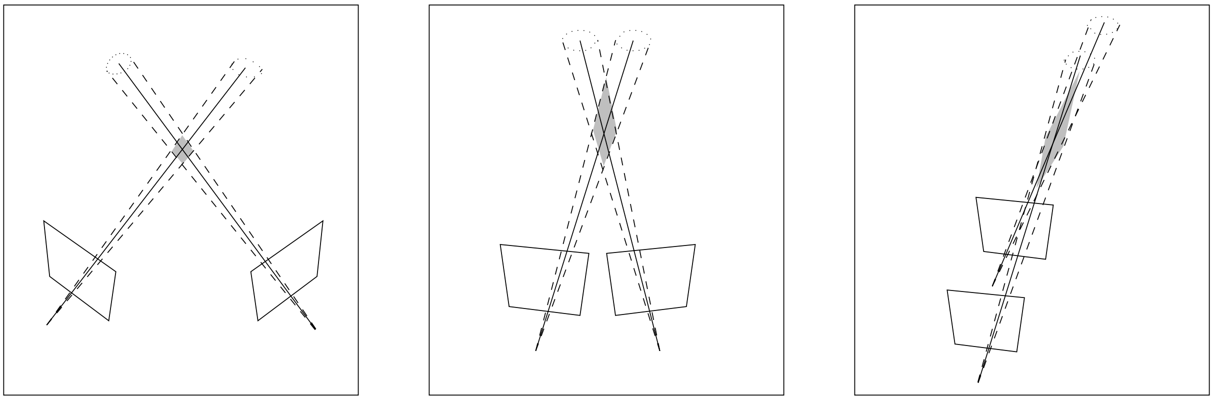


Figure 3.7: In each subfigure the shape of the grey shaded region illustrates the uncertainty of the reconstruction. The obtainable accuracy is obviously dependent on the angle between intersecting rays. The uncertainty is increased the sharper the angle become. Owing to almost parallel rays, especially forward motion yields poor reconstruction results [17].

3.5 Multiview Reconstruction

3.5.1 Bundle Adjustment

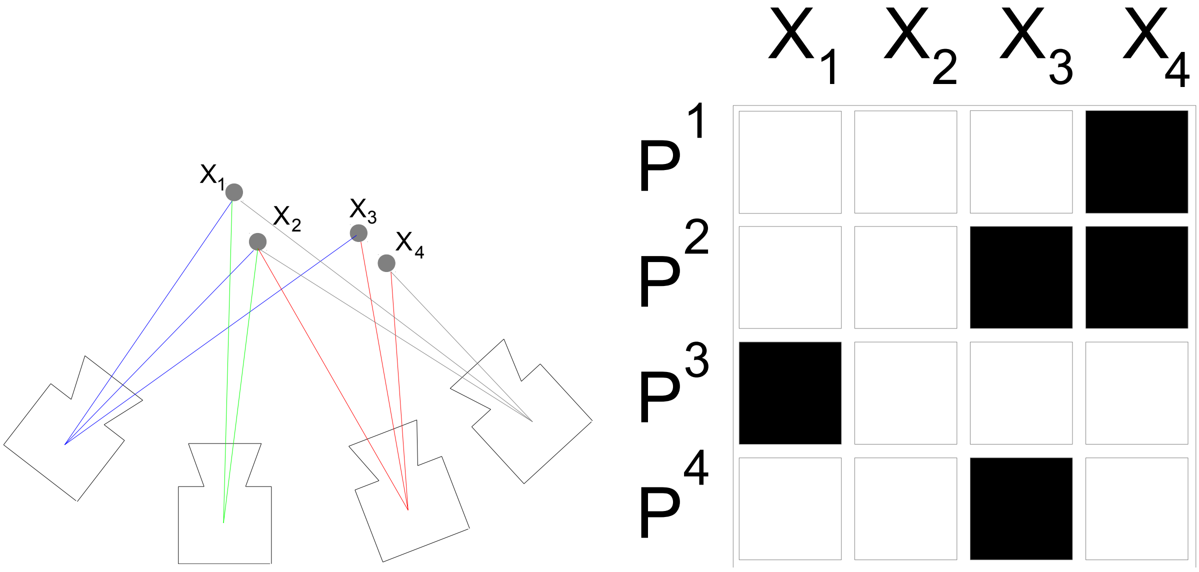
In the previous section the reconstruction of a 3D point using a triangulation algorithm was described. Due to noise in the measured image coordinates x and x' the epipolar constraint $x'^T Fx = 0$ cannot be satisfied exactly. Furthermore, it was assumed that accurate camera matrices are available. Especially in SLAM this assumption cannot be guaranteed and the accuracy of the reconstructed map does not only depend on the measured point correspondences, but also on the camera poses.

Therefore an algorithm is necessary that simultaneously optimizes the coordinates of all 3D points and all the parameters of the camera matrices. This can be achieved by the concept of bundle adjustment (BA), which was originally developed in the field of photogrammetry in the late 1950's by Duane C. Brown [6].

In a mathematical framework, a set of 3D points $\mathbf{X}_j$ is given which is visible in a set of cameras with projection matrices P^i. If the j-th point $\boldsymbol{X}_j$ is seen by the i-th camera the measured image coordinates are denoted as $\boldsymbol{x}_j^i$. Due to noise in the image measurements and non optimal poses, the equations $\mathbf{x}_j^i = P^i\mathbf{X}_j$ cannot be satisfied exactly. Hence, the aim is to find camera matrices $\hat{P}^i$ and 3D points $\hat{\mathbf{X}}_j$ that minimize the reprojection error:

$$\min_{\hat{P}^i,\hat{\mathbf{X}}_j} \sum_{i,j} d(\hat{P}^i\hat{\mathbf{X}}_j, \mathbf{x}_j^i)^2, \tag{3.16}$$

where $d(\boldsymbol{x}, \boldsymbol{y})$ denotes the Euclidean distance between homogeneous points $\boldsymbol{x}$ and $\boldsymbol{y}$.
Since bundle adjustment provides the Maximum Likelihood (ML) estimate in the presence of Gaussian noise [17], the algorithm is almost always used as a final step in the reconstruction process. In addition, BA is tolerant to missing image projections.



	X_1	X_2	X_3	X_4
P^1	□	□	□	■
P^2	□	□	■	■
P^3	■	□	□	□
P^4	□	□	■	□

Figure 3.8: A typical bundle adjustment problem consisting of four points X_1 to X_4 and four cameras. Due to the scene structure or errors during matching, only a subset of all points is visible in each image. The information whether a point is visible in a specific camera or not is usually represented in a connectivity matrix. If an entry is white, a point is visible in the camera, otherwise it is not.

Unfortunately, bundle adjustment is usually a extremely large optimization problem. Since each projection matrix has 11 parameters (six from the exterior orientation and five from the internal orientation) and each 3D point has 3 degrees of freedom (DOF) often thousands of parameters have to be optimized. For example, in a typical SLAM environment the map consists of several thousand points and a few hundred camera poses. Therefore, not only a minimal parametrization is needed but also an implementation which avails the special structure of BA.

Furthermore, the relationship between the measurements and the parameters is nonlinear and a direct solution cannot be found. Instead, the cost function of equation (3.16) has to be minimized iteratively given an initial parameter estimate that can be obtained by the methods described in chapter 3.4.

In order to obtain the ML estimate, bundle adjustment is formulated as a huge non-linear least squares problem.

Therefore, all parameters from the projection matrices P^i and all 3D points $\mathbf{X}_j$ are put in a joint parameter vector $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_n)$ in Euclidean space $\mathbb{R}^n$. All image measurements $\mathbf{x}_j^i$ are converted to the measurement vector $\mathbf{M}$ in Euclidean space $\mathbb{R}^m$ and the functional relation between $\mathbf{M}$ and $\boldsymbol{\beta}$ is denoted by $\mathbf{M} = \mathbf{f}(\boldsymbol{\beta})$.

In the case of bundle adjustment the number of measurements m is greater than the number of parameters n $(m > n)$ and the equation $\mathbf{M} = \mathbf{f}(\boldsymbol{\beta})$ cannot be satisfied exactly. Hence, the aim is to find a parameter vector $\hat{\boldsymbol{\beta}}$ for which $||\boldsymbol{\epsilon}||$ in the functional relation $\mathbf{M} = \mathbf{f}(\hat{\boldsymbol{\beta}}) - \boldsymbol{\epsilon}$ is minimized.

Now the cost function has to be minimized. Unfortunately, real cost functions are very complicated and cannot be minimized in closed form. Instead, it is approximated at the current parameter estimate $\boldsymbol{\beta}$ via a Taylor expansion and the aim is to find a displacement $\boldsymbol{\beta} \rightarrow \boldsymbol{\beta} + \boldsymbol{\Delta}$ that minimizes the approximate local model

$$h(\boldsymbol{\beta} + \boldsymbol{\Delta}) \approx h(\boldsymbol{\beta}) + g^T \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta}^T H \boldsymbol{\Delta},$$

where $h(\boldsymbol{\beta})$ is a arbitrary scalar-valued function, g the gradient vector of h

$$g = \frac{\delta h}{\delta \boldsymbol{\beta}}(\boldsymbol{\beta})$$

and H is the Hessian matrix

$$H = \frac{\delta^2 h}{\delta^2 \boldsymbol{\beta}}(\boldsymbol{\beta}).$$

With the assumption that H is positive definite, $h(\boldsymbol{\beta})$ is a strictly convex function with a unique global minimum. The minimum of this function with respect to $\boldsymbol{\Delta}$ is found by differentiating $h(\boldsymbol{\beta})$ and setting the derivative to zero:

$$H\boldsymbol{\Delta} + g = 0,$$

or

$$H\boldsymbol{\Delta} = -g. \tag{3.17}$$

Starting from some initial estimate $\boldsymbol{\beta}_{\mathbf{0}}$, which can be calculated with the methods described in chapter 3.4, the parameter increments $\boldsymbol{\Delta}$ can now be iteratively calculated until convergence. This procedure is called Newton's method and is known for its exceptionally rapid convergence if the cost function is approximately quadratic near the minimum [57].

Unfortunately, the second derivatives H have to be calculated, which is not only implementally demanding but also computationally expensive. Furthermore, Newton's method may not converge to a minimum but rather to a saddle point [57]. By having a closer look at the cost function that has to be minimized in bundle adjustment the function may be written as

$$h(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\epsilon}(\boldsymbol{\beta})\|_2^2 = \boldsymbol{\epsilon}(\boldsymbol{\beta})^T\boldsymbol{\epsilon}(\boldsymbol{\beta})/2.$$

The error vector $\boldsymbol{\epsilon}(\boldsymbol{\beta}) = \boldsymbol{f}(\boldsymbol{\beta}) - \boldsymbol{M}$ is given by

$$\boldsymbol{\epsilon}(\boldsymbol{\beta}) = \begin{pmatrix} \epsilon_1(\boldsymbol{\beta}) \\ \vdots \\ \epsilon_m(\boldsymbol{\beta}) \end{pmatrix},$$

where $\epsilon_j(\boldsymbol{\beta})$ calculates the difference between the j-th 3D point $\boldsymbol{X}_j$ projected by the i-th camera P^i and the measured image point $\boldsymbol{x_j^i}$:

$$\epsilon_j(\boldsymbol{\beta}) = P^i\boldsymbol{X}_j - \boldsymbol{x}_j^i.$$

The gradient vector $g(\boldsymbol{\beta})$ is then $g(\boldsymbol{\beta}) = \boldsymbol{\epsilon_\beta}\boldsymbol{\epsilon} = J^T\boldsymbol{\epsilon}(\boldsymbol{\beta})$, where J represents the Jacobian matrix and subscripts denote differentiation. In order to obtain the Hessian H, the gradient vector is differentiated a second time which leads to

$$H = \boldsymbol{\epsilon}_{\boldsymbol{\beta}}^T\boldsymbol{\epsilon}_{\boldsymbol{\beta}} + \boldsymbol{\epsilon}_{\boldsymbol{\beta\beta}}^T\boldsymbol{\epsilon}.$$

Ignoring the second derivatives gives the so-called Gauss-Newton method and the Hessian can be calculated via $H = \boldsymbol{\epsilon}_{\boldsymbol{\beta}}^T\epsilon_{\boldsymbol{\beta}} = J^TJ$. Now equation (3.17) can be rewritten by the normal equations to yield

$$J^TJ\boldsymbol{\Delta} = -J^T\boldsymbol{\epsilon}. \tag{3.18}$$

The shift vector $\boldsymbol{\Delta}$ is then given by solving the normal equations as discussed in [17].
The main advantage of this method is that the complicated Hessian can be approximated by a product of the first derivatives J^TJ. However, this is only valid for a linear function $\boldsymbol{f}(\boldsymbol{\beta})$ and a disadvantage of the Gauss-Newton method is its poor convergence if this assumption does not hold and the discarded terms are not negligible [57].

Another method to calculate the shift vector $\boldsymbol{\Delta}$ is gradient descent. Let $-g = -\boldsymbol{\epsilon}_{\boldsymbol{\beta}}^T\boldsymbol{\epsilon}$ be the negative gradient vector, then it defines the direction in which the cost function decreases the most.
Starting from an initial estimate $\boldsymbol{\beta_0}$ the shift vector is calculated iteratively by

$$\lambda\Delta = -g_{\boldsymbol{\epsilon}}. \tag{3.19}$$

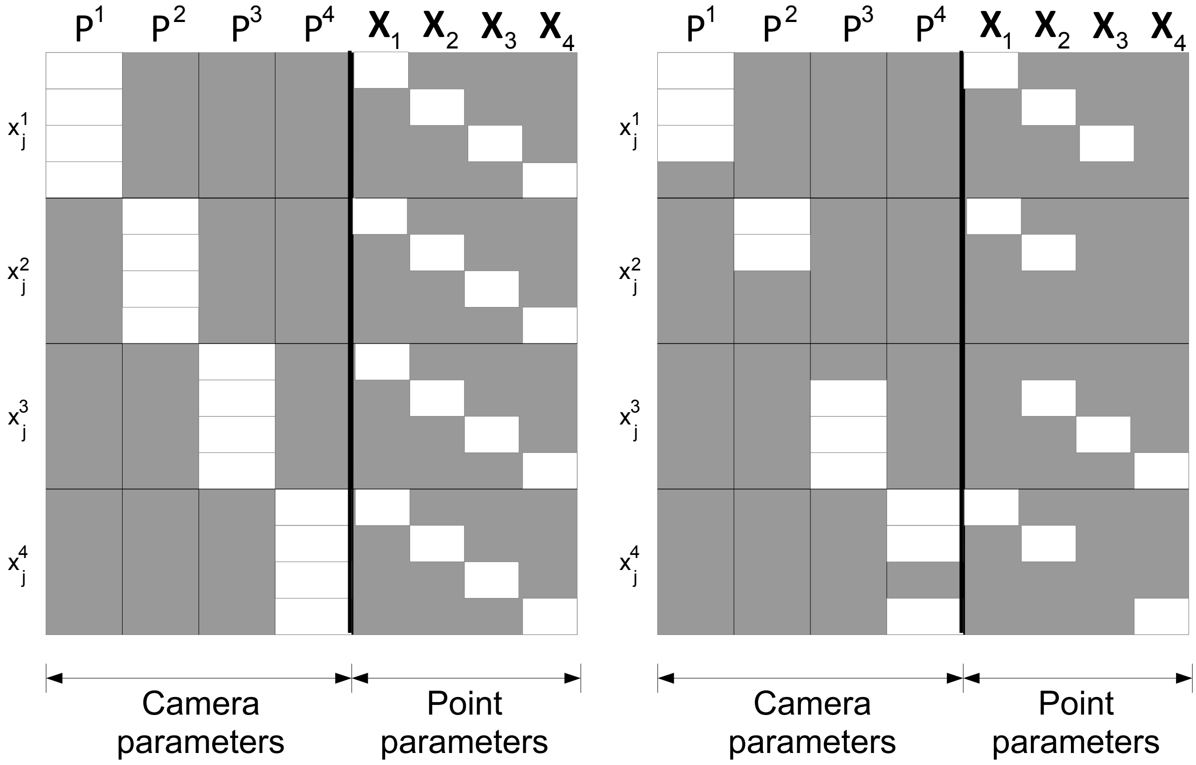
The parameter λ defines the length of the step and can be chosen adaptively. However, in order to obtain the minimum λ has to be small, which usually leads to a slow convergence. Therefore, the gradient descent itself is not recommended for iterative minimzation.
Nevertheless, it is used in combination with Gauss-Newton yielding the Levenberg-Marquardt method (LM), which is commonly used for bundle adjustment. Here the shift vector $\mathbf{\Delta}$ is given by

$$\mathbf{\Delta} = -(J^T J + \lambda I)^{-1} J \boldsymbol{\epsilon}, \tag{3.20}$$

where λ is a damping factor that varies in each iteration.
If $\lambda = 0$ the LM-algorithm turns into the Gauss-Newton method (3.17), whereas for $\lambda \to \infty$ the shift vector is calculated by the gradient descent (3.19).
In expectation of a rapid convergence λ is typically initialized with a small value. In each iteration the value is adjusted depending on the shift vector $\mathbf{\Delta}$ calculated by equation (3.20). If the shift vector reduces the error function then the improved parameter vector is maintained and λ is divided by a factor (a common factor is 10). Otherwise the recently computed parameter vector is discarded, λ is multiplied by the preset factor and equation (3.20) is solved again with the old parameter estimate until the error function is reduced.

An issue of the LM algorithm is its huge complexity. The dominant computational cost of the algorithm is the solution of the normal equations (3.18) in each iteration. Since this has complexity N^3 in the number of parameters N the algorithm is only suitable for least-squares minimization problems with a few parameters. Since bundle adjustment optimizes thousands of points and hundreds of camera poses simultaneously, the implementation as stated above is not recommended.
Fortunately, the normal equations in bundle adjustment exhibit a sparse block structure which can be exploited by the LM-algorithm resulting in the sparse Levenberg-Marquardt algorithm. Therefore, the parameter vector $\boldsymbol{\beta}$ is partitioned into two parameter vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ such that $\boldsymbol{\beta} = (\boldsymbol{a}^T | \boldsymbol{b}^T)^T$. In bundle adjustment the vector $\boldsymbol{a} = (a_1^T, a_2^T, ..., a_i^T)$ contains the parameters of the camera matrices P_i and $\boldsymbol{b} = (b_1^T, b_2^T, ..., b_j^T)$ the parameters of the 3D points $\boldsymbol{X}_j$.
The division of the parameters leads to a block structured Jacobian $J = [A|B]$, where the submatrix A is the Jacobian with respect to the camera parameters $\boldsymbol{a}$ and submatrix B the Jacobian with respect to the 3D points $\boldsymbol{b}$.

This block structure of the Jacobians is shown on the left of figure 3.9.



(a) (b)

Figure 3.9: Form of the Jacobian structure of bundle adjustment. Figure 3.9a shows the situation in which each map point $\boldsymbol{X}_j$ is visible in each camera P^i. Figure 3.9b shows the form of the Jacobian for the example of figure 3.8. Only if a map point is visible in a certain camera the entry has to be calculated. Otherwise it is zero.

Now the normal equations (3.18) can be written as[1]:

$$\left[\begin{array}{c|c} A^TA & A^TB \\ \hline B^TA & B^TB \end{array}\right] \begin{pmatrix} \boldsymbol{\Delta_a} \\ \hline \boldsymbol{\Delta_b} \end{pmatrix} = \begin{pmatrix} A^T\boldsymbol{\epsilon} \\ \hline B^T\boldsymbol{\epsilon} \end{pmatrix} \tag{3.21}$$

The next step is the augmentation of the matrix in equation (3.21) by multiplying their diagonal entries with the factor $1 + \lambda$. The resulting matrices may be rewritten as $(A^TA)^* = U^*$ and $(B^TB)^* = V^*$. Hence, equation (3.21) alters to:

$$\begin{pmatrix} U^* & W \\ W^T & V^* \end{pmatrix} \begin{pmatrix} \boldsymbol{\Delta_a} \\ \boldsymbol{\Delta_b} \end{pmatrix} = \begin{pmatrix} \epsilon_A \\ \epsilon_B \end{pmatrix}.$$

---

[1] Note, that it is assumed that the covariance matrix is the identity matrix

After a left multiplication of both sides by $\begin{pmatrix} I & -WV^{*-1} \\ 0 & I \end{pmatrix}$ the top right block vanishes resulting in

$$\begin{pmatrix} U^* - WV^{*-1}W^T & 0 \\ W^T & V^* \end{pmatrix} \begin{pmatrix} \boldsymbol{\Delta_a} \\ \boldsymbol{\Delta_b} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\epsilon}_A - WV^{*-1}\boldsymbol{\epsilon}_B \\ \boldsymbol{\epsilon}_B \end{pmatrix}. \tag{3.22}$$

Now the shift vector $\boldsymbol{\Delta_a}$ for the camera parameters can be computed by solving the top half of equation (3.22):

$$(U^* - WV^{*-1}W^T)\boldsymbol{\Delta_a} = \boldsymbol{\epsilon}_A - WV^{*-1}\boldsymbol{\epsilon}_B.$$

The shift vector for the point parameters $\boldsymbol{\Delta}_b$ can be computed by back-substitution of $\boldsymbol{\Delta}_a$ in

$$V^*\boldsymbol{\Delta}_b = \boldsymbol{\epsilon}_B - W^T\boldsymbol{\Delta}_a$$

.

In the Jacobian matrix of figure 3.9a it was assumed that each point is visible in each camera. In real world applications this is normally not the case and the sparse LM-algorithm can be easily adapted to this situation by setting the relevant entries to zero. The resulting Jacobian for the example given in figure 3.8 can be verified in figure 3.9b.
In this thesis no further attention is given to this and the interested reader is invited to consult the detailed book *Multiple View Geometry in computer vision* of Hartley and Zisserman [17] for implementation details and Triggs et al. [57] for a deeper comprehension.

3.5.2 Robust Bundle Adjustment

Bundle adjustment gives the Maximum Likelihood (ML) solution under the assumption of Gaussian noise by minimizing the reprojection error (3.16) in a least-squares sense. However, the main drawback of least-squares is its high sensitivity to outliers.

Instead of minimizing a cost function, a parameter vector $\boldsymbol{\beta}$ can also be obtained by maximizing the likelihood function p.

$$\hat{\boldsymbol{\beta}} = \arg\max(p(\epsilon_1, ..., \epsilon_n) = \prod_{i=1}^{n} f(\epsilon_i)), \tag{3.23}$$

where $\hat{\boldsymbol{\beta}}$ is the optimal parameter vector, $p(\epsilon_1, ..., \epsilon_n)$ is the probability of a set of n measurements with errors ϵ_i and $f(\epsilon_i)$ denotes the probability distribution of the i-th error.
Equation (3.23) can be rewritten by taking the negative logarithm

$$\hat{\boldsymbol{\beta}} = \arg\min(-\log(p(\epsilon_1, ..., \epsilon_n)) = -\sum_{i=1}^{n} \log(f(\epsilon_i))). \tag{3.24}$$

Under the assumption that the data exhibits a Gaussian distribution the Probability Density Function (PDF) is given by $p(\epsilon) = \exp(-\epsilon^2)$. Substituting the PDF in equation (3.24) leads to a cost function $C(\epsilon)$

$$C(\epsilon) = \epsilon^2, \tag{3.25}$$

which is the cost function used in least squares (see figure 3.10).

Unfortunately, the squared error is not robust to outliers. Due to the quadratic form of the curve, outliers gain a vast influence on the optimization result.

Therefore, Huber [19] proposed the replacement of the function $-\log(f(\epsilon_i)))$ by a symmetric positive definite function $\rho(\epsilon_i)$ that has a unique minimum.

This alters equation (3.24) to

$$\hat{\boldsymbol{\beta}} = \arg\min \sum_{i}^{n} (\rho(\epsilon_i)). \tag{3.26}$$

Equation (3.26) can be minimized by differentiating $\rho(\cdot)$ and setting the derivative to zero:

$$\sum_{i=1}^{n} \psi(\epsilon_i) \frac{\partial \epsilon_i}{\partial \beta} = 0 \tag{3.27}$$

where $\psi(x) = \frac{\partial \rho(x)}{\partial x}$ is called the influence function of p.

Based on the influence function a weight function w can be defined by

$$w(x) = \frac{\psi(x)}{x}$$

and equation (3.27) alters to

$$\sum_{i=1}^{n} w(\epsilon_i) \epsilon_i \frac{\partial \epsilon_i}{\partial \beta} = 0. \tag{3.28}$$

This equation system can efficiently be solved using the Levenberg-Marquardt algorithm as discussed in chapter 3.5.1. The influence of the weights is considered in the calculation of the shift vector $\boldsymbol{\Delta}$ by

$$\boldsymbol{\Delta} = -(J^T w J + \lambda I)^{-1} w J \boldsymbol{\epsilon}, \tag{3.29}$$

where w is a diagonal weight matrix that is adjusted in each iteration.

Several M-estimators have been introduced in literature assuming different probability distributions or influence functions.

The most common estimators are Huber's M-estimator and Tukey's M-estimator [19].

The Huber function is given by

$$\rho(x) = \left\{ \begin{array}{cc} \frac{x^2}{2} & x < k \\ k(x - \frac{k2}{2}) & x \geq k \end{array} \right. .$$

As shown in figure 3.10 the Huber cost function has a quadratic form for a small error x and increases linearly if the error exceeds a threshold k. As a concequence, the Maximum Likelihood is obtained, whereas the effect of outliers is reduced by assigning them a decreasing weight. Another commonly used M-Estimator is Tukey's biweight [19] which is defined as

$$\rho(x) = \begin{cases} \frac{k^2}{6}(1 - [1 - (\frac{x}{k})^2]^3)) & x \leq k \\ \frac{k^2}{6} & x > k \end{cases}.$$

This M-Estimator is a so-called cut-off estimator because residuals above a threshold k get zero weight and do not influence the optimization result (see figure 3.10). This is in contrast to Huber's M-estimator in which each residual is assigned a weight $w > 0$.

In order to obtain an asymptotic efficiency on the standard normal distribution both M-Estimators require the choice of a constant threshold k. Setting k to a fixed value is not recommended since the residuals usually decrease during the iterative minimization. Instead, k is calculated in each iteration dependent on the variance σ of the errors. A common choice for the threshold is $k = 1.345\sigma$ for the Huber M-Estimator and $k = 4.6851\sigma$ for Tukey's, respectively.
The variance σ is usually not calculated from the dataset directly, but instead a robust variance estimator based on the median of the errors is used. Throughout this thesis the variance estimator as proposed by Rousseeuw [40] is used

$$\sigma \approx 1.4826(1 + \frac{5}{m-n})\sqrt{\tilde{x}}, \tag{3.30}$$

where $\tilde{x}$ is the median of the squared error vector, m is the length of the error vector and n is based on the dimensionality of the residuals.
The procedure in robust bundle adjustment is described in the following: At first, the residual vector is computed. The next step is to calculate the median $\tilde{x}$ of the squared vector. Based on the median the robust variance is obtained using equation (3.30). Subsequently, the threshold k is calculated and each residual is then assigned a weight by evaluating the weight function. Next, the shift vector $\boldsymbol{\Delta}$ is calculated using equation (3.29) followed by an update of the parameter vector $\boldsymbol{\beta}$. These steps are repeated iteratively until convergence occurs.

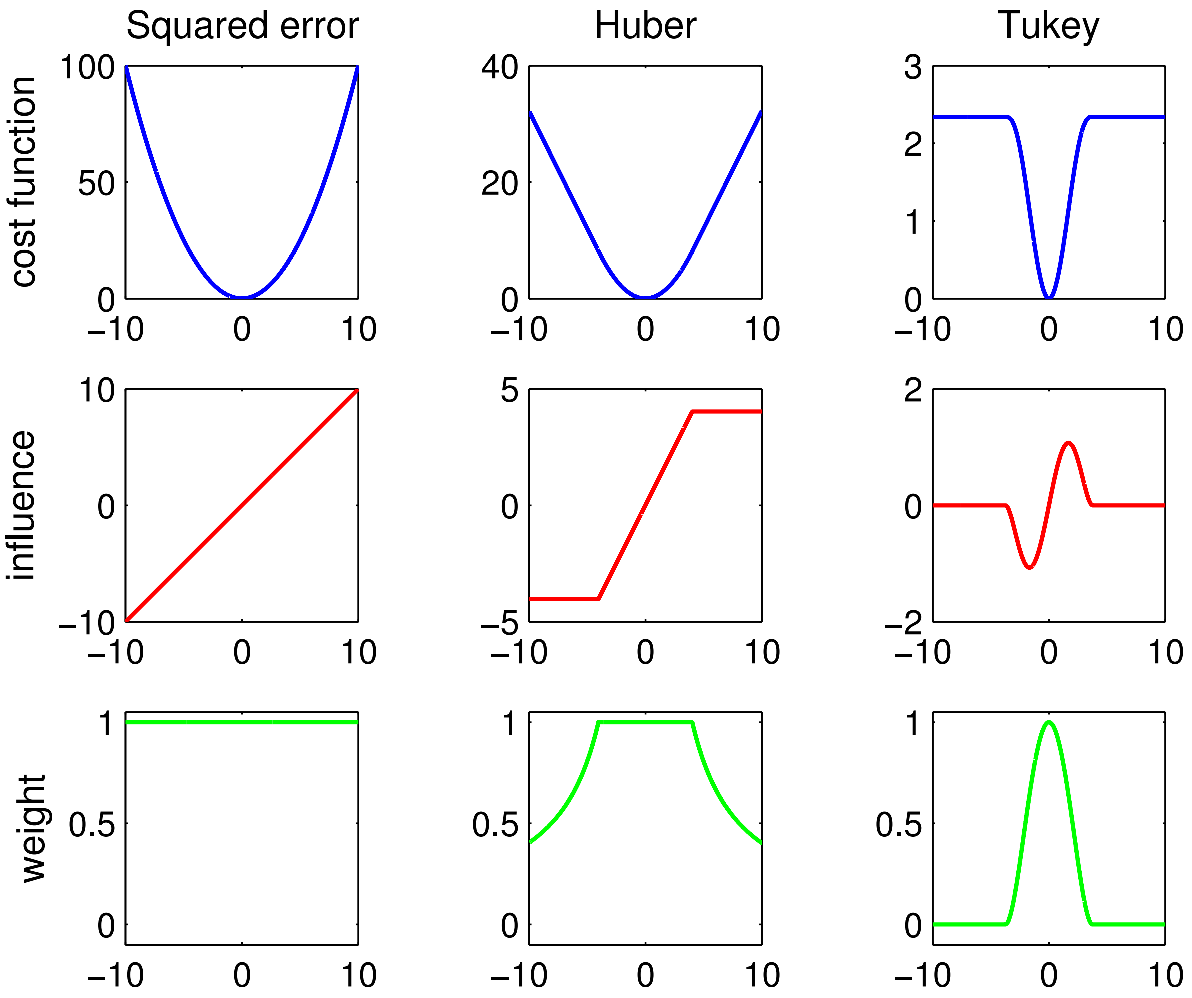


Figure 3.10: A comparison of three different cost functions and their attendant influence and weight functions. Due to the quadratic form of the error function in least-squares, outliers have a huge influence on the optimization result. In contrast, the Tukey and the Huber M-Estimator reduce this influence by assigning distant outliers less weight in the optimization process.

CHAPTER 4

Proposed Method

This chapter provides a general overview of the proposed method which is closely related to the seminal work of Klein and Murray [22] named parallel tracking and mapping (PTAM). However, only the most basic structure of the system as well as its fundamental differences to PTAM are described here. Readers who are familiar with Klein's work may refer to the subsequent chapters 5 and 6 for additional details.

The implemented method comprises two different approaches: The first is almost identical to PTAM and based on FAST-10 corners and patch-based matching. The other one replaces the FAST corners by Difference of Gaussian (DoG) keypoints and SIFT features. Beside these changes, the entire system is essentially the same in both cases and will be described in the following paragraph.

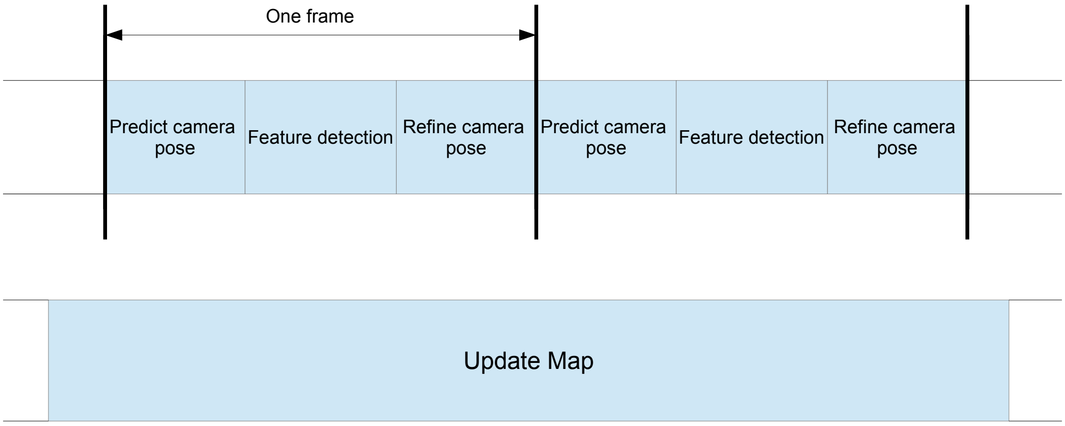


Figure 4.1: Tracking and mapping is split into two different tasks. The tracker is responsible for estimating the camera pose at every frame. The map is only updated at certain points in time.

The fundamental idea of both suggested approaches is to split tracking and mapping into two different subtasks as it is shown in figure 4.1. On the one hand there is the tracking thread. Every time a new frame is acquired this thread is responsible to estimate the current camera pose. Therefore, the pose is first predicted based on a motion model. Subsequently, a vast amount of keypoints is detected in each frame. Finally, the camera pose is refined using a reconstructed three dimensional model of the observed scene.

On the other hand there is the mapping thread that has to create this model. However, not every frame contributes to the map. Instead only a small subset of heuristically chosen frames, the so-called keyframes are selected for mapping.

Obviously there is some sort of interaction between the two threads. The mapping thread provides the map for the camera pose estimation, whereas the tracker occasionally drops keyframes for map extensions. In contrast to PTAM the multi-core support has been discarded and instead the two threads are executed alternatingly.

Furthermore, both threads have different temporal requirements. At every incoming frame the tracker has to instantaneously determine the camera pose. Therefore, this thread should be real-time capable. By contrast, since not every frame is used for mapping this thread can spend lots of time creating a rich and accurate map. An overview of the entire system is shown in figure 4.2.

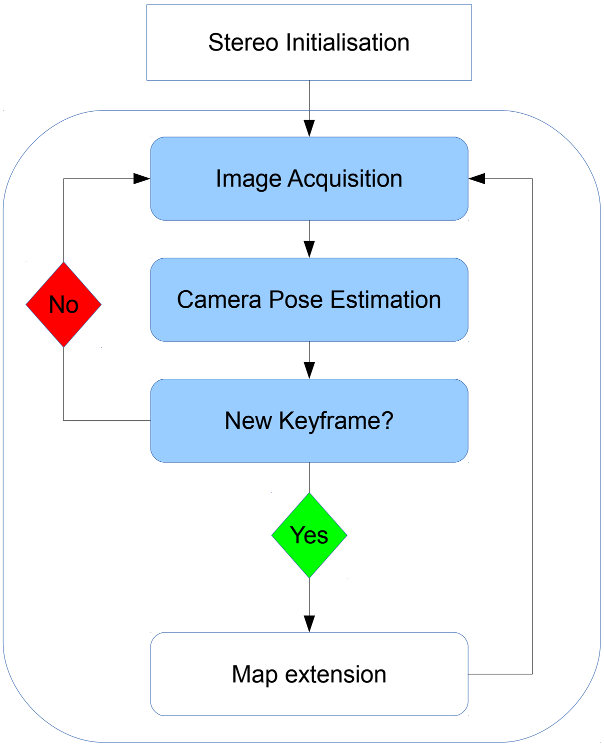


Figure 4.2: Overview of the proposed SLAM system. White rectangles indicate the tasks of the mapping thread, whereas blue rectangles symbolize tasks that are entirely up to the tracker. The system is first initialised using a stereo reconstruction algorithm. Whenever a new frame is acquired, the camera pose is estimated using the reconstructed map. From time to time, the map is extended by storing a new keyframe and new map points.

At first an initial map has to be built employing a stereo algorithm. Thereto the user points the camera at the workspace to be reconstructed and takes an image. This image is maintained in the system as the first keyframe. Subsequently, the camera has to be translated to a slightly displaced position where the second keyframe is acquired.

Both views are then used to create a cloud of point features. Therefore, the suggested method provides two different approaches that have to be chosen in the forefront.

The first one is identical to PTAM and detects FAST corners in a four level image pyramid of both keyframes. The second one replaces the corners by DoG-keypoints. Depending on the preset choice, point correspondences between both views are either found by patch-based matching in the case of FAST or SIFT feature matching in the case of the latter. All found matches are then used to obtain a robust estimate of the essential matrix E. Both camera matrices are extracted from E and the initial map is triangulated with the remaining point correspondences. Finally, the camera poses and all reconstructed map points are immediately refined through robust bundle adjustment.

Once the map is initialized, the system enters an infinite loop: Whenever a new frame is acquired, the camera pose is predicted based on a motion model. Subsequently, all existing map points are projected into the current frame and identified using either small patches or SIFT-features. Found 3D-2D correspondences are then used to refine the six degree of freedom camera pose by minimizing a robust cost function of the reprojection error.

The initial map only covers a relatively small area and has to be extended as soon as the camera leaves the mapped area. Thereto, the tracker gives the current frame to the mapping thread where it is stored as a keyframe.

Since the tracker has assigned a camera pose and a set of keypoints to each keyframe, new 3D points can easily be added through triangulation with another keyframe that is already contained in the map. Correspondence search is again performed employing either patch-based or feature-based matching. Remaining correspondences then become new map points. Once the triangulation is done, the entire map and all camera poses are refined using robust bundle adjustment.

CHAPTER 5

Localization

In general simultaneous localization and mapping (SLAM) has to answer two questions: “Where am I?” and “What is the structure of my surrounding?”
This section will give the answer to the first of these fundamental questions: “Where am I?” Or in other words: It is assumed that an accurate map is given a priori and that this map is used to estimate the position of a robot or sensor relative to the existing map.
This process is called localization and is one of the most important problems on the way to a fully autonomous robot, since many subsequent tasks such as obstacle avoidance require a precise position. According to Thrun et al. [55] there are three different types of localization problems that can be distinguished by their difficulty and amount of initial knowledge:

Local Localization
Local localization or pose tracking is the simplest of the three tasks because it is assumed that the initial robot pose is known. Whenever the robot or sensor moves, a tracking system ensures that the pose is determined accurately.

Global Localization
In the absence of an initial pose global localization has to be performed. This problem occurs whenever an off-state robot is switched on and has to localize itself from scratch. Because of that the problem is often called wake-up problem. Since the initial position is completely unknown, the pose error is not bounded and hence global localization is more difficult than the local one. As soon as the global localization is completed the robot can switch to pose tracking.

Kidnapped Robot Problem
The kidnapped robot problem is closely related to global localization but presents an even harder task. It occurs whenever an operating robot is kidnapped and moved to a new position. Now

one might argue that the task is simply global localization but with the essential difference that the robot believes to know where it is although it does not. The scenario of a kidnapped robot is rather odd but in practice it occurs very often however in a slightly different way:
Regardless of how well the quality of a tracking system is, under certain circumstances tracking will fail. In visual SLAM for example rapid camera movements cause motion blur that will render patches useless for tracking. Therefore an autonomous system has to detect whenever tracking has failed in order to take action for recovering from these failures and prevent the map from being corrupted.

Another influence on the difficulty of tracking is the behaviour of the environment. In general two different kinds of environments can be considered: Static environments and dynamic environments.
In static environments nothing except the robot is moving. As a consequence the environment does not change over time and once a map is created it is always up to date. Furthermore the robot does not have to consider other moving objects while making decisions.
Unfortunately a static environment is only an approximation to the real world which is not static at all. All objects may change their positions over time and maps can be outdated within moments after their creation. Moreover the robot has to keep observing the world and involve other moving objects while deciding upon an action.
Since tracking in dynamic environments is much harder than in static ones, the main focus of the following chapter is on the latter. However, in visual SLAM inevitable changes such as variations of light will occur, and it is necessary to implement some methods that give resilience to such changes.

5.1 Local Localization

In the following, it is assumed that a map as mentioned in chapter 6 is given and that the initial pose of the camera is known. Every time a new frame is acquired from the hand-held camera a tracking system has to estimate the pose of the camera relative to the existing map.
In general a pose describes the actual position of the camera relative to a world coordinate frame and can be minimally parametrised as a six-dimensional vector. The first three elements usually comprise the translation along the x, y and z axes and the other three the rotation around these axes relative to the WCS. The entire tracking process is shown in figure 5.1 and will now be discussed in detail.
The pose of a camera at time t is encapsulated in the projection matrix P_t

$$P_t = K[R_t|\boldsymbol{t}_t] = KC_t, \tag{5.1}$$

where the notation is the same as in chapter 3.1. The aim is to determine the external parameters R_t and $\boldsymbol{t}_t$. The internal camera parameters in K are assumed as being fixed and hence not dependent on time t.

The local localization consists of a two-stage tracking procedure:

- Estimation of a prior pose C_t based on C_{t-1} (Prediction step)
- Refinement of the pose (Measurement update)

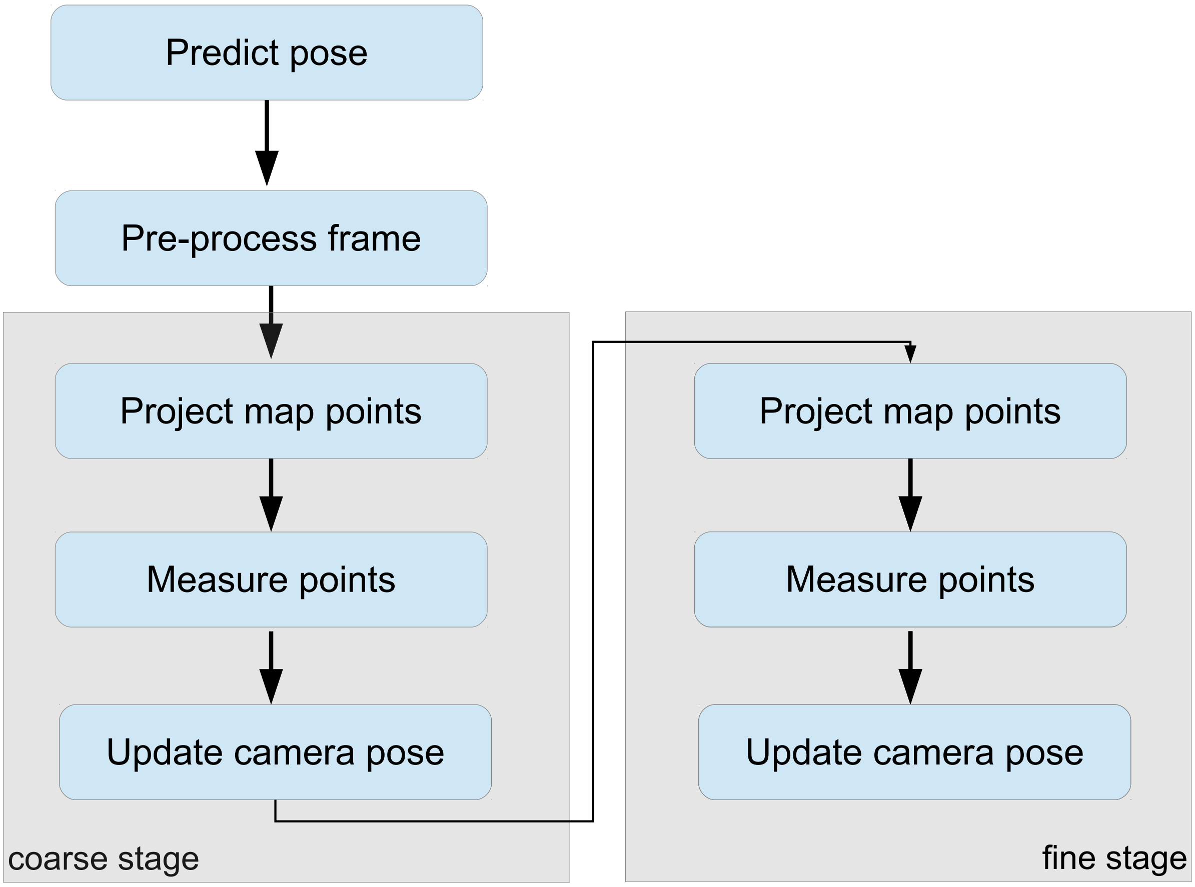


Figure 5.1: A predicted pose is calculated based on a motion model. Subsequently, keypoints are detected in the current frame. Existing map points are projected into the image and identified using their attendant descriptors. Found 3D-2D correspondences are then used to refine the camera pose. Note that the coarse to fine stage is only possible for the FAST approach.

5.1.1 Prediction

In this step the camera position is predicted based on assumptions or empirical values. A good prediction simplifies and accelerates the subsequent measurement step.

The easiest assumption would be a static camera with a pose C_t equal to the previous pose C_{t-1}. This static model yields satisfying results for most of the autonomous SLAM scenarios where a slowly moving robot is equipped with a camera. However, this model cannot cope with fast camera motions and tracking eventually fails.

Another method to predict the actual pose is to use a decaying velocity model as proposed in [22]:

The prior pose C_t is given by

$$C_t = M_C C_{t-1},$$

where the inter-frame motion M_C can be calculated by the exponential map using the velocity estimate v_{t-1} and the time step Δt:

$$M_C = \exp v \Delta t.$$

Subsequently, the velocity is updated by using the 6 DoF motion vector μ obtained by the tracker:

$$v_t = 0.9 * (v_{t-1} + \mu).$$

The presented motion model has the advantage that it provides camera tracking despite large velocities under the assumption of moderate accelerations. However, in situations in which the user observes an interesting object he may decelerate or suddenly stop moving. This unexpected change in speed cannot be represented by the underlying model and in the worst case tracking fails. By contrast, a constant velocity model would easily cope with this situation.

5.1.2 Pose Refinement

Regardless of how the prior pose was estimated it is based on assumptions of a possible camera motion. If the movement differs from the corresponding prediction the prior pose is only a rough estimate of the true pose. Therefore it has to be refined using the existing map. This procedure will be first described for the patch-based approach and subsequently for the SIFT approach.

At first the recently acquired frame is pre-processed and a four-level image pyramid is constructed, subsampling the original image three times by a factor of two. Then in each level a huge number of FAST-10 corners [39] is detected.

The following steps may be divided into two parts: A coarse stage in which only a small number of low-level points is projected into the frame, followed by a fine stage in which points of all pyramid levels are used. In both stages the procedure is the same:

After projecting the points into the image a fixed-range patch search is performed in order to identify correspondences between 3D map points and recently detected FAST corners. Effectively compensating large scale changes the system automatically decides in which level of the image pyramid a specific map point should be searched. Resulting 3D-2D correspondences are then used to refine the camera pose by minimizing a robust objective function of the reprojection error.

It should be emphasized that the coarse stage is optional and is only used to increase resilience against rapid camera motions. In SLAM sequences with a slowly moving robot and high framerate, the coarse stage is not mandatory because the prior pose is usually close to the actual one. However, in hand-held SLAM this assumption cannot always be guaranteed and the prior pose might be far away from the true pose. Refining the prior pose first with around 50 points of pyramid level three and four is not very time consuming but usually gives a much better initial estimate for the non linear optimization in the subsequent fine-stage.

The procedure in the case of the SIFT approach is very similar. However, there is no need for creating an image pyramid and correspondence search is performed over the entire image. Furthermore, a coarse-to-fine tracking is not possible and all map points are projected at once.

Project map points

After estimating a prior pose the three dimensional map points are projected into the image plane. The procedure is the same as described in chapter 3.2. Since the map points are represented in world coordinates they first have to be transformed to the camera coordinate system and then further projected into the image plane. These steps are summarized by

$$\hat{\boldsymbol{x}} = P_t \boldsymbol{X}_w, \tag{5.2}$$

where $\boldsymbol{X}_w$ denotes the map points presented in the WCS, P_t the 4x4 projection matrix at timestep t and $\hat{\boldsymbol{x}}$ the two dimensional image coordinates in the IPCS. The set of map points $\boldsymbol{X}_w$ can be further divided into a potentially visible set (PVS) $\boldsymbol{X}_{wv}$ if their projected coordinates lie within the image borders and map points that cannot be seen from the current view because they are projected outside the image.

The next step is to find the exact location of each map point $\boldsymbol{X}_{wv}$ in the image. This is achieved by comparing the patch that is associated with each map point with the patches extracted at the FAST-10 corner positions. In order to reduce computational effort the search is limited to a fixed-range around the predicted location of the point. The influence of illumination changes is reduced by using the normalized cross-correlation (NCC) as a similarity measure. Furthermore a map point is only considered as found if the correlation is above a threshold.

In contrast to PTAM no affine warp is performed on the extracted patch to compensate changes in perspective because it is based on assumptions that cannot always be fulfilled. Nonetheless, the patch warp was examined during the implementation stage but usually less 3D-2D correspondences were found than without patch warping. However, in order to handle large scale changes, PTAM uses the warping matrix to determine the pyramid level in which a certain map point has to be searched. Therefore, the need for a different criterion arises.

This is done by transforming each map point to the camera coordinate system of the keyframe in which it was first observed. Then the Euclidean norm of each point is calculated. The map points are already transformed to the CCS of the actual frame by the prior pose and the Euclidean norm is computed as well. The ratio between the norm of a specific point in the CCS of the keyframe and the norm of the same point in the actual frame is now used to decide in which pyramid level the search should be performed. If this ratio is higher than a certain threshold the patch is searched in the next pyramid level. As soon as the ratio exceeds an even higher threshold a search in a higher level is performed. This criterion was tested in a sequence with forward camera motion yielding more 3D-2D correspondences.

The procedure for the SIFT approach is quite similar to the one described above but since the features are scale-invariant there is no need for an image pyramid and only the original image is used for feature detection.

Pose update

After identifying the map points in the current view the prior pose can be updated by minimizing the reprojection error using the found 3D-2D correspondences:

$$\min_{\hat{P}_t} \sum_{j \in S} d(\boldsymbol{x}_j, \hat{P}_t \boldsymbol{X}_j)^2, \tag{5.3}$$

where S is now the subset of the PVS that were successfully matched.

Unfortunately, outlier free correspondences cannot be guaranteed. Thus, minimizing the reprojection error in a least-squares sense in the presence of outliers has a great influence on the optimization result and thus on the estimated camera pose. In order to provide some robustness to outliers, a minimization of a robust objective function of the reprojection error is performed which alters equation (5.3) to

$$\min_{\hat{P}_t} \sum_{j \in S} \rho(d(\boldsymbol{x}_j, \hat{P}_t \boldsymbol{X}_j)), \tag{5.4}$$

where $\rho(\cdot)$ is the Tukey biweight objective function [19]. The used parameters can be found at the end of section 3.5.1.

As one can see, the equation above is effectively the same as equation (3.16) used in robust bundle adjustment and can be minimized with the techniques described in section 3.5.1. However, the essential difference is that the map points are excluded from the optimization and assuming that the intrinsic camera parameters K are known a priori, only the parameters of the 6-DoF pose have to be optimized.

5.2 Discussion

In this chapter the localization of a camera was discussed given an accurate map. At first three kinds of localization problems were distinguished each with a different amount of prior knowledge and increasing difficulty. In order to solve the first problem, two approaches have been suggested with some advantages but also some limitations that will be examined in the following.

Since the local localization of the camera has to be done at every frame, the underlying method should be real-time capable. This can be achieved by the patch-based approach in combination with FAST corner detection since the computational costs are menial. FAST in general outperforms many state-of-the-art keypoint detectors in terms of reliability, quantity and especially speed [39]. Thus, depending on the coding language, hundreds of salient image points can be detected in a few ms. The extraction of small patches around each keypoint and the subsequent data association using a simple correlation based approach can be done at high frame-rate as well.

Furthermore, the usage of a simple motion model speeds up the tracking process by providing a prior camera pose with little computational effort. The prior pose is then used to project the map points into the actual frame. Under the assumption that the predicted pose approximates the true pose well enough, the search range for the subsequent identification of map points can be restricted to a small circular region around the projected point. This reduces the number of necessary patch comparisons and makes the system operate at full frame rate.

However, if the camera motion is not in conformity with the predicted movement the map points may be projected far away from their true position. Due to the limited search range, 3D-2D correspondences may not be found yielding a tracking failure. As a consequence, the choice of the underlying motion model in combination with the preset search range is a critical part of the patch-based approach. A large search region compensates a wrong motion model but slows down the tracking system whereas a small search region increases the speed of the correspondence search but tracking may fail due to an inaccurate prior pose.

By far the weakest part of the suggested approach is the patch-based correlation. Each map point is assigned a patch extracted from the keyframe in which it was first observed. After projecting a map point into the actual frame it is searched by checking its patch against patches in the local neighbourhood of the projected location. As a similarity measure, the NCC is used in order to gain some resilience to linear illumination changes. However, NCC is not tolerant to nonlinear lighting changes and huge camera rotations. Furthermore, no actions to compensate perspective changes due to different points of view are taken into account.

In order to allow some variations in the appearance of a patch and to filter inevitable mismatches a preset threshold is used. Nevertheless, the value of the threshold is a critical factor. Setting the threshold to a low value makes the NCC more tolerant to view point and illumination changes but yields more mismatches. In contrast, a high threshold decreases the chance of false matches but many false positives may be discarded as well.

As a concequence, the choice of the “right” threshold looms large in the robustness and accuracy of the system. However, a few wrong matches can be handeld by the system because in the subsequent pose refinement a robust objective function of the reprojection error is minimized and mismatches are flagged as outliers by the M-Estimator.

Up to now the suggested approach cannot handle large scale changes. If the camera moves forwards or backwards the patch of a map point covers a different area and cannot be matched. To overcome this problem, a four-level image pyramid is used and based upon a criterion the system decides in which level a certain map point has to be searched. However, the threshold that decides whether the search of a point should be performed in another pyramid level is difficult to choose. Since usually sufficient map points are found during the correspondence search this is not really a problem but performance and robustness may be decreased a bit.

Another downside of the patch-based matching is accuracy. The 3D-2D correspondences are only found with pixel precision. In PTAM an inverse compositional approach [3] was proposed but owing to real-time constraints this could not be done for every patch. In literature [47] various others approaches exist but none of these were implemented in this thesis.

Many of the problems described above can be solved with the second approach suggested in this thesis. Since the employed SIFT features are robust against illumination and viewpoint changes, they can usually be matched despite large variations in appearance. Another advantage of this approach is the descriptor's invariance against image scale and rotation. Since keypoints are detected in a scale-space no additional criterion is needed where a specific map point has to be searched and scale changes can be hand-held much easier than in the patch-based approach. Furthermore, the SIFT-algorithm provides sub-pixel accurate locations of the keypoints which will result in a higher accuracy.
However, these improvements will be reflected in the computation time. The detection of keypoints lags behind the performance of the FAST detector and the subsequent extraction and matching of the descriptors will also take more time than the patch-based approach [39]. Additionally, each descriptor is compared with all descriptors extracted in the current frame and the search is not restricted to a small area around the predicted location. However, the SIFT algorithm usually finds significantly less points than the FAST detector which reduces the number of necessary comparisons.

Despite fixing some issues of the patch-based approach there are still some problems both methods cannot cope with. Regardless of how well the motion model predicts the prior camera pose, fast accelerations cannot be handled. This is mainly due to the arising of motion blur. In both approaches, motion blur will either cause that no keypoints can be found in the image or that the descriptors are rendered useless for correspondence search.
To overcome this problem Klein et al. improved their method PTAM by using small edgelets. These are combined with the FAST corners and support the system to keep track of the camera despite fast motions and accelerations. Adding edgelets to the system remains for future work. Another approach is suggested by Newcombe et al. [36]. The camera is tracked in real-time by a dense map. In a comparison with PTAM this results in a more robust tracking performance especially during fast camera motions.
Neither the methods proposed in this thesis nor the improvements discussed above can handle images without texture. Although keypoints might be extracted, the subsequent matching step will most certainly fail because the descriptors cannot be distinguished and data association will fail.
This is a general problem of vision based-approaches and probably the only way to overcome these limitations is in combination with other sensors that are not exposed to this kind of failure.

CHAPTER 6

Map Building

In the last chapter it was shown how a robot can keep track of its position if an accurate map of the environment is available. Unfortunately, in many cases like for example for mars explorations a map cannot be provided at all. Sometimes an a priori map may exist in terms of a computer-aided design (CAD) model which was blueprinted by an architect. These models are often not up to date and even worse, they cannot meet the requirements visual approaches usually need for accurate tracking. Even if the model was rich enough to support the robot's tracking, it would assume a static environment without any artificial objects included. Hence, it cannot support the robot in choosing the right actions in terms of evading obstacles or interacting with its surrounding. Moreover, an operation in a highly dynamic environment such as road traffic is inconceivable.

In order to accomplish these tasks an up-to-date map is required all the time and therefore has to be created from scratch. This would be a relatively easy task if an oracle told the robot its true position. However, this is clearly not possible. A method for robot localization was given in the previous chapter but this unfortunately requires a map. As a concequence, the robot is situated in some kind of a chicken-egg problem: On the one hand it has to construct a map in order to make tracking possible and on the other hand mapping can only brought off if an accurate pose estimate is provided. The only way to solve this problem is to simultaneously build a map while localizing itself in an unknown and previously unmapped area. This is quite difficult as the robot has to deal with inaccurate poses and noisy sensor measurements. Therefore, a solution to this problem is necessary that estimates the "best" map and the "best" robot position given the noisy sensor data.

As a concequence of the topics discussed above, the map architecture is one of the core parts of a SLAM system and in the first place it depends on the used sensor and the focus of the SLAM approach. Furthermore, it has to fulfill several requirements to make the robot's tracking and its interaction with the observed environment as accurate as possible.

Since the focus of this thesis is on visual SLAM one might argue to use all images since they contain the most information and do not need further treatment. However, many images overlap and thus yield redundant information. In addition, storing each image is memory and time consuming. Thus, a trade-off between attention to detail, memory consumption and practical employment has to be found.

This can be achieved by employing the keyframe concept. A keyframe is a snapshot that is occasionally pulled out of the incoming video stream. Only the amount of information contained in the keyframe is used for map representation. Probably the most common representation of SLAM is a map that consists of features. Features in visual SLAM are mostly points [22] or lines [23] and can be seen as a sparse representation of the whole scene. As mentioned in chapter 3.3 the features are typically distinctive and salient such that they can characterize the observed scene in a sufficient and preferably unique way. Since it is very difficult to find a specific point or line in an image the features are usually assigned a descriptor. This descriptor can either be a small part of the image around the keypoint or it describes the local properties around the point. With these descriptors it is possible to establish a relation between the map and the current frame.

As shown by [22] this representation is adequate for most of the tasks that occur in monocular SLAM as it is not only detailed but also memory efficient. However, in high-level tasks such as obstacle avoidance or interaction with the environment a dense reconstruction of the observed scene is necessary.

6.1 Map Architecture

This chapter describes the architecture of the map M as used in chapter 5. It is mainly motivated by the system of Klein and Murray [22]. At first, the architecture of the patch-based approach is described and subsequently the differences to the SIFT approach are discussed.

As mentioned above, using each image is redundant and optimizing over all measurements and visited poses soon becomes too time consuming. Hence, only a subset of all frames, the so-called keyframes F_n, are maintained and build the map information.

$$M = \{F_1, F_2, ..., F_n.\}$$

Associated with each keyframe is a camera-centred coordinate system C_n that describes the camera pose relative to the world coordinate system W. Furthermore, for each keyframe a four-level image pyramid L_{ik} is constructed:

$$F_i = \begin{cases} C_i & \\ L_{ik} & k = 1,2,3,4 \end{cases}$$

In the world coordinate system there are also several three dimensional points. The coordinates of the j-th point in the WCS are denoted $\boldsymbol{X}_{jW} = (X_{jW}, Y_{jW}, Z_{jW}, 1)$. Then a map M can be described as follows:

$$M = \begin{cases} F_n & n = 1, 2, ..., N \\ \boldsymbol{X}_{jW} & j = 1, 2, ..., J \end{cases}$$

Each map point $\boldsymbol{X}_j W$ also has a reference to a keyframe which is typically the one in which the point was first observed. This is stored together with a reference to the pyramid level of observation and the pixel coordinates $\boldsymbol{x}$ in the first pyramid level and yields to

$$\boldsymbol{X}_{jW} = \begin{cases} F_i \\ L_{ik} \\ \boldsymbol{x} \\ n(\cdot) \end{cases},$$

where $n(\cdot)$ denotes a patch that is assigned to each map point. The patch is extracted around the point's location in its keyframe and pyramid level and contains the raw gray values. The patch is not stored explicitely but extracted whenever it is needed.
Additionally, a binary $N \times J$ visibility matrix of all keyframes and all map points exists. The visibility matrix indicates if a map point $\boldsymbol{X}_j W$ is visible in a keyframe F_n.

In this thesis a second approach based on SIFT-features is used. The architecture of the map remains essentially the same but a few trifles alter. Due to the scale-invariance of the descriptor there is no need of an image pyramid and a keyframe F_i only consists of the original image L_i and the camera pose C_i:

$$F_i = \begin{cases} C_i \\ L_i \end{cases}.$$

Furthermore, a map point is not represented by a patch but by a SIFT descriptor $d(\cdot)$:

$$\boldsymbol{X}_{jW} = \begin{cases} F_i \\ L_i \\ \boldsymbol{x} \\ d(\cdot) \end{cases}.$$

6.2 Map Initialization

As mentioned in chapter 3.4.2 the uncertainty of reconstruction depends on the angle between intersecting rays. Since a monocular camera is used throughout this thesis initializing the map can only be done by triangulating corresponding points of sequentially acquired images. Hence, moving forward at the beginning would yield a very uncertain scene geometry. To overcome this problem user interaction is required by taking a snapshot of the observed scene followed by a translation and an optional rotation of the camera. After completion another snapshot is taken and these two views become the first keyframes of the system. Since the map additionally needs 3D map points the two images are further used to initialize the map. The subsequent steps are slightly different for both approaches. According to customs, the procedure for the patch-based approach is described first.

Initially, a four level image pyramid is constructed for both keyframes and in each level FAST-10 corners are detected. Around each corner position 11x11 pixel patches are extracted and correspondences are found using normalized cross correlation (NCC). The putative matches are then used to compute the relative orientation between the two views. This is done by calculating a robust estimate of the essential matrix via the five or eightpoint algorithm and RANSAC. Without further knowledge the first camera is assumed to be at the origin of the world coordinate system with corresponding projection matrix $P_1 = K[I|0]$. The projection matrix of the second view P_2 can then be derived from the essential matrix.

Both projection matrices and the remaining point correspondences can now be used to triangulate the three dimensional map points as described in 3.4.2. In order to obtain the maximum likelihood estimate of the map and the camera poses both are immediately optimized through bundle adjustment as discussed in chapter 3.5.1. To reduce the influence of outliers a robust objective function of the reprojection error is minimized. The underlying objective function is Tukey's biweight.

The map initialization is essentially the same for the SIFT approach. However, no image pyramid is necessary and instead of FAST corners DoG keypoints are detected. The correspondence problem is then solved by matching SIFT features.

Although the intrinsic camera parameters are assumed to be known, the overall scale of the initial map cannot be determined. In [22] it is obtained by assuming a camera translation of 10cm between the first two frames. However, this thesis starts from the premise that a calibration plate with known size is available and thus the scale can be determined. If this is not the case the scale remains an undetermined variable throughout the sequence.

6.3 Map Extension

The initial map consists of two keyframes and between 300 and 400 map points. Thus, it only covers a small area. As soon as the camera leaves the mapped region no or only a few map points can be identified in the actual frame and tracking will fail sooner or later. Therefore, the map has to be extended by adding new keyframes and points, whenever the sensor explores an unmapped territory. In order to prevent the map from being corrupted as it was often the case in filtering methods, keyframes are only added when certain requirements are met: Keyframes are only added to the map if the tracking quality is considered as sufficiently good. In order to decide whether tracking is good or bad the amount of matched features is used as an indicator. If the number of found 3D-2D correspondences is above a certain threshold tracking is considered as good. Furthermore, this indicates that the acquired image is not exposed to motion blur. However, if many map points are used for tracking the camera might also be in a sufficiently mapped area and thus keyframe insertion is not required.

To counter this the camera has to be traveled a certain distance from the position where the last keyframe has been acquired. The advantage of this condition is twofold. On the one hand it prevents the map of being corrupted because a sufficient baseline between two views is guaranteed. On the other hand it ensures that there is still enough overlap between the two views in order to create new map points.

Once these conditions are met and the system decides to acquire a new keyframe all map points that were not used for tracking are projected into the image a second time. This is done in order to identify as much map points as possible. If a map point is observed successfully its entree in the visibility matrix is altered, whereas the rest is considered as not visible.

The next step is to extend the map by adding new points. Each pyramid level of the keyframe is already equipped with a huge number of FAST corners that were detected during the tracking stage. However, the vast amount of corners is usually not necessary and the set is reduced by discarding all corners that are beneath the mean of the corner score. This ensures, that only the most salient points of an image are retained. Next, all corners within a small circular region around each projected map point are discarded as well. This prevents a blob-like clustering of map points.

All remaining FAST corners can potentially become new map points. Since triangulation is not possible from one view another frame is required. A good choice is the latest keyframe acquired since not only sufficient overlap but also an adequate baseline is guaranteed. In order to find point correspondences, 11x11 pixel patches are extracted at the corner's location in both images. As a similarity measure normalized cross correlation is used and only matches with values above a certain threshold are considered as found. However, the search is only performed on the same pyramid level. These putative matches are then used to obtain a robust estimate of the essential matrix. This is a bit waste of time since the essential matrix is not examined further but usually the correspondences are almost outlier free.

The procedure for the SIFT approach is exactly the same. Only the FAST corners are replaced by DoG keypoints and the patch-based matching by SIFT feature matching. As a threshold the nearest neighbour is used.

All remaining point correspondences are now used to triangulate new map points. This is done by the simple triangulation algorithm of 3.4.2. The necessary projection matrices are the ones that are assigned to both keyframes.

Subsequently, all map points and all poses except the first one, which is fixed are then bundle adjusted as described in chapter 3.5.1. In order to be robust against outliers the Tukey M-Estimator is used with the parameters given in chapter 3.5.2.

Unfortunately, full bundle adjustment has a computational complexity of $O(N^3)$ for N keyframes. If the map size is small the bundle adjustment complexity can be reduced to $O(N^2M)$ for N keyframes and M map points but real-time exploration will soon become impossible.

6.4 Discussion

In this chapter, the map used in the suggested approaches has been introduced. Since it is not only responsible for accurate tracking but also to support real-time exploration and many high-level tasks, the map architecture is one of the core parts of the SLAM system.
The architecture of the map arose during the examination of the former state-of-the-art approaches that were based on filtering methods. These methods intermeshed tracking and mapping and whenever a new frame is acquired the camera pose and all features in the map have to be updated together. As a concequence, every frame contributes to the map and failures in data association immediately reflect in a permanent corruption of the map. Thus, researches had to make huge efforts to prevent this from happening.
The breakthrough succeeded by splitting tracking and mapping into two different threads. As soon as these two tasks are probabilistically independent, the map does not have to be updated at every frame. Instead, only if a frame meets certain requirements it is chosen to contribute to the map. These frames are then denoted keyframes and only their amount of information is represented in the map. In contrast to the previously prevailed filter methods this concept depicts a great improvement. By restricting the whole available data to only a small but distinctive subset of the incoming video stream the amount of redundant information is diminished. Furthermore, if not every frame is used for map building, the mapping thread does not have to operate at full frame rate. Instead, it can spend much time to extract as much information as possible and make the map as accurate as possible. Additionally, a map corruption can be prevented if the system gets a feedback about tracking quality. If tracking difficulties have been detected the actual frame might not be suitable for map extension and is not considered by the mapping thread.
Although this architecture has many advantages, there are also some limitations. Probably the most fundamental one is the drop rate of keyframes since it contributes a lot to exploration speed and tracking success. If keyframes are dropped too often, the amount of redundant information increases and even worse the map scales poorly with the number of keyframes. This soon prevents the system from real-time exploration. On the other hand if keyframes are dropped too rarely, image overlap decreases and adding new map points may become impossible. Furthermore, the choice of keyframes is based on heuristics that have to be detected from the tracking thread. If this detection fails the current frame may be unsuited for map extension and possibly yields a irretrievable map corruption.

Beside the architectural issues, additional problems that can be ascribed to the usage of a monocular camera and the proposed patch-based approach occur. The first problem arises at the reconstruction of the scene. This is only possible if the same scene is observed from different point of views and the uncertainty of a reconstructed point is determined by the angle between the intersecting rays. Hence, a problem occurs if the camera movement is purely forwards or backwards. After the camera has traveled a certain distance adding a new keyframe and new map points is obliged. If this is not done, the patch-based approach fails because correspondences are only searched in the same pyramid level. Due to the (almost) parallel rays this yields an uncertain scene geometry. Note, that this problem is decreased in the SIFT approach since correspondences are found in a scale-space.

Unfortunately, due to the map architecture additional problems occur that neither the patch-based nor the SIFT-based approach can handle. The map optimization is done by the highly accurate but computationally demanding bundle adjustment (BA). Although this has not to be done every frame since mapping is separated from tracking, BA scales poorly with the map size. Full bundle adjustment of the whole map and all keyframes N has a computational complexity of $O(N^3)$. Hence, with increasing map size real-time exploration soon becomes impossible.

CHAPTER 7

Experiments

As mentioned in chapter 1, the goal of this thesis is the development of a SLAM system that can construct an accurate map in previously unknown areas while providing a robust estimate of the current camera pose in real-time. Therefore, the proposed method has been evaluated in various datasets. The first one is the New College Vision and Laser Data Set [45]. It is a typical SLAM sequence gathered outside by a slowly moving robot. The main focus of this evaluation is in terms of tracking accuracy, resilience against lighting changes and scale-drift.

The second dataset is a plaster reproduction of the Temple of Dioskouroi famously known as the “Temple” [42]. Although this is not a typical SLAM dataset but rather the yardstick for multi-view stereo reconstruction algorithms, the sequence has been used to evaluate the attainable map quality.

In addition to the external recorded datasets, a sequence with a hand-held monocular camera has been acquired in a small office environment. Since, no ground truth is available the sequence starts at a well defined point and after a few hundreds of frames it returns to exactly the same point. Therefore, the sequence is ideally suited to demonstrate the ability of the system to close small loops.

The following chapter is organized as follows: At first an overview over each dataset along with required pre-processing steps is given. Then, the datasets are evaluated using the patch-based approach first and subsequently the feature-based approach. Finally, both methods are compared in terms of robustness, accuracy and computation time.

7.1 Datasets

In the following, each dataset is presented in detail. Thereto, the image acquisition process is depicted and required pre-processing steps are explained.

7.1.1 New College Dataset

The New College Vision and Laser Data Set consists of 30 GB data gathered within the college's grounds and adjoining parks in November 2008. The dataset is divided into three epochs and covers a distance of 2.2 km. It is a pure outdoor dataset with multiple loop closures and mainly used for large-scale SLAM. An aerial view of the whole dataset is shown in figure 7.1.

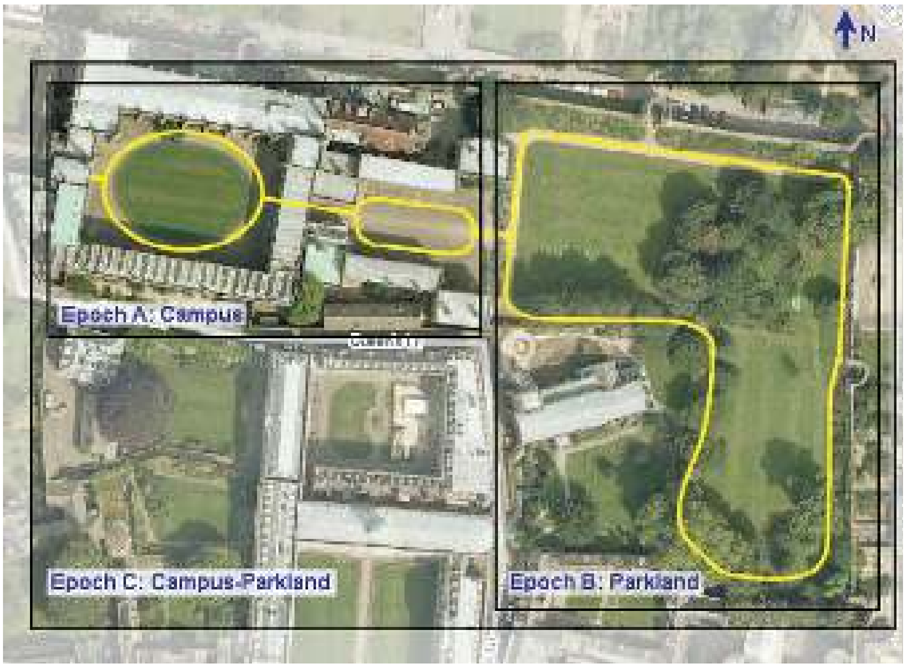


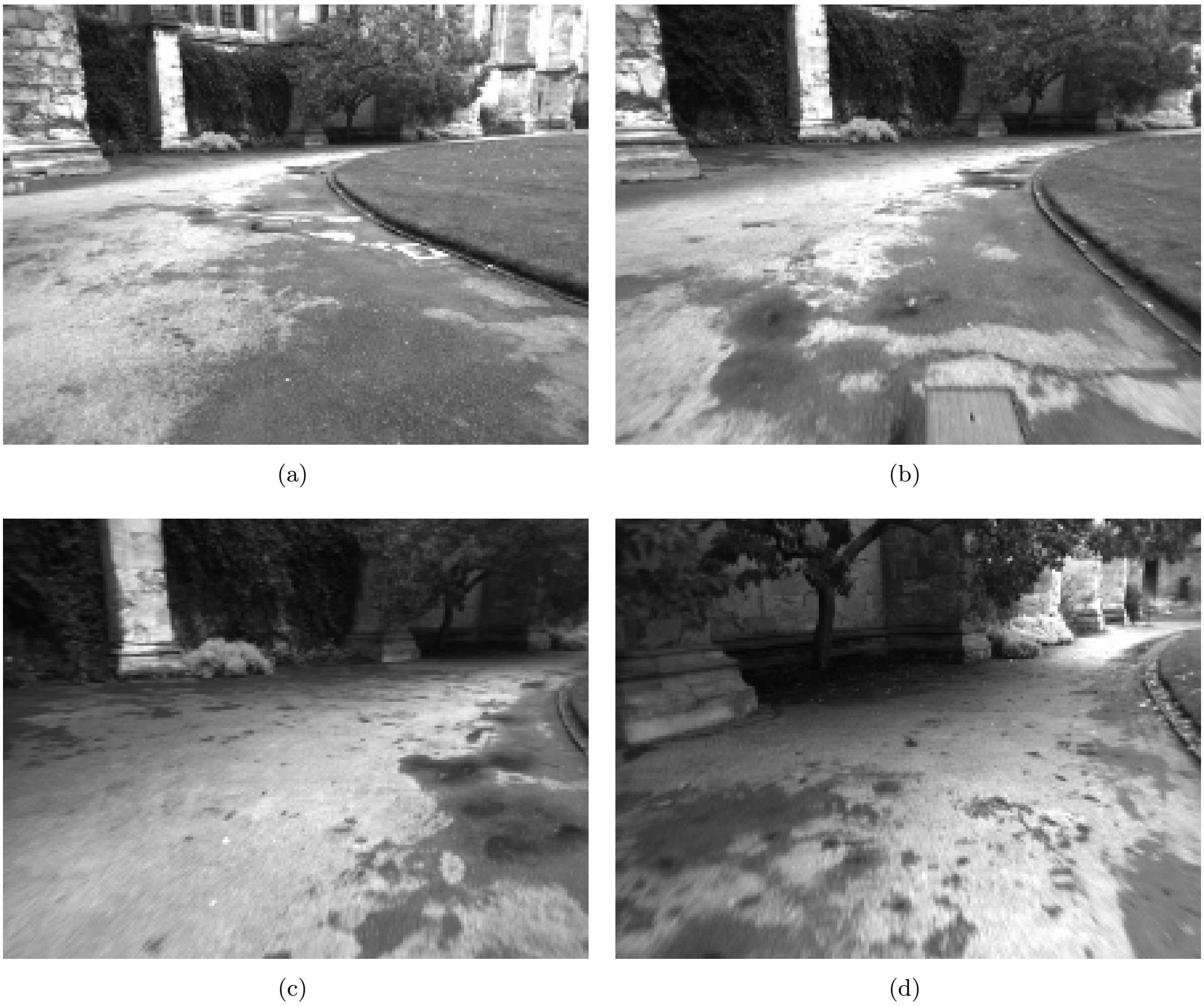
Figure 7.1: Aerial view of the New College Vision and Laser Data Set. Driven paths are marked in yellow. For evaluation purposes a small part of the Epoch B: Parkland has been selected [45].

The whole data set was recorded by the robot Lisa which is equipped with two laser scanners, a stereo camera, a panoramic camera and a GPS receiver. Since this thesis is focused on visual SLAM, only the 512x384 grayscale images of the stereo camera are used. These images are on average captured at 20 Hz and all camera parameters are available.

However, the provided sequence contains only the raw images as captured by the stereo rig and thus exhibit lens distortions. Since the proposed method expects a removal of those, all images are undistorted and rectified using the standalone C++ source file provided by the authors.

Furthermore, the direction of the coordinate axis are contrary to the convention of the proposed method. For evaluation purposes the two coordinate systems were aligned yielding a consistent comparability.

In order to do justice to the monocular approach, only the images of either the left or the right camera are used. However, the initial map is always constructed using the corresponding images of the stereo rig. This is not only done to guarantee a sufficient baseline, but also to obtain a metric scaling for subsequent ground-truth comparisons. After initialization the system solely makes use of the images of one camera.


(a) (b)

(c) (d)

Figure 7.2: Snapshots of the New College Dataset. (a) shows the first frame of the selected sequence. (b) After a few tens of frames the robot has purely moved forward. (c) The robot enters the curve. (d) The robot has almost reached the end of the curve.

Since the entire dataset would blow the scope of a small area, only a subset of it within the Parkland has been selected. This choice has been made, because this part consists not only of forward motions but also of a rambling curve and scale changes.

The sequence starts with a static camera and subsequently passes over into a slow but almost pure forward motion. After approximately 100 frames an increasing movement in the y-direction can be registered. This is due to the starting curve. There is also a small motion in the x-direction throughout the whole sequence but this is almost negligible compared to the remnant movements.

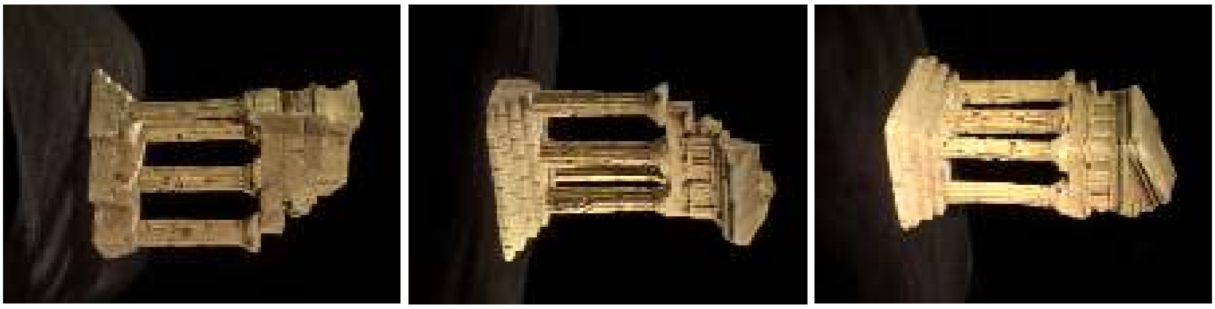
The images are taken at a high frame rate by a slowly moving robot and thus, view point changes between consecutive frames are marginal. Furthermore, the images rarely exhibit motion blur and contain lots of texture needed for keypoint detection. As illustrated in figure 7.2 some parts of the images show few structure and the illumination is not constant.

However, the most challenging part for the proposed monocular approach is the forward motion at the beginning of the sequence since this yields a poor reconstruction of the map.

7.1.2 Middlebury Dataset

The Middlebury dataset has been published by Seitz et al. [42] in 2006 and provides, for the first time, a huge database of calibrated multi-view stereo images and accurate ground-truth. The dataset is publicly available and can be downloaded at the attendant homepage [41] which is continuously updated.

The dataset comprises two different sequences of which each is further divided into three sub datasets. For evaluation purposes, the “TempleRing” has been selected. It contains 47 images taken on a ring around the object. Each image was captured by a CCD camera and has a resolution of 640x480 pixels. The intrinsic and extrinsic camera parameters are available for every view and the distortions are already removed. During the acquisition process the camera did not move and thus, the images do not exhibit motion blur. However, they are unordered and view point changes between consecutive images are very large. This made it impossible to use the entire ring and the greatest connected component of 22 images is used for evaluation. A selection of three images is shown in figure 7.3.



(a) (b) (c)

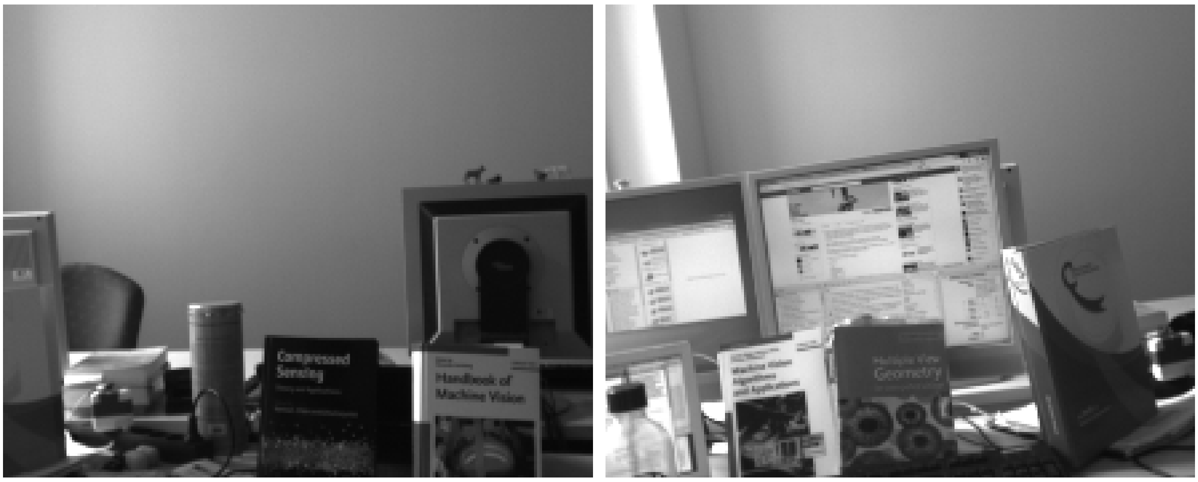
Figure 7.3: Snapshots of the Middelbury Dataset.

7.1.3 Office Scene

The office scene is a small sequence of approximately 500 frames captured on average at 13Hz. The greyscale images have a resolution of 1024x1280 pixels but are resized to 512x640 pixels in order to decrease processing time.

Since the sequence is self recorded, some pre-processing steps were required. At first, the calibration matrix and the radial distortion coefficient were determined by imaging a calibration grid from 25 points of view. Subsequently, all images were undistorted and rectified.

The sequence starts with a static camera at a well defined point within the office. After a few frames, the camera slowly begins to move in the y-direction. After approximately 215 frames the camera turns back and is finally placed at the starting point.



(a) (b)

Figure 7.4: Snapshots of the Office Dataset.

Due to slow motions, the images exhibit very little motion blur. However, most of the images contain a huge amount of repetitive texture as it can be verified in figure 7.4. This exacerbates keypoint extraction and feature matching.

7.2 Evaluation

In the following, the evaluation of the three sequences is described. At first, the patch-based approach is examined in terms of accuracy, robustness, and scale-drift. In respect of a subsequent comparison the same experiments are performed using the SIFT approach as well.

7.2.1 Evaluation Using Patches

At first, the patch-based approach is evaluated on the New College dataset. Due to the usage of the stereo image pair for map initialization, the reconstruction works well and after retaining only the most salient corners, the initial map consists of approximately 300 points. Since view point changes are marginal between consecutive images, a huge amount of these can be identified in each incoming frame. Usually between 120 and 230 three dimensional points can be found in each image. Thereof, approximately 55% impinge on the first pyramid level, 30% on the second, 10% on the third and only 5% on the fourth pyramid level. However, the threshold of the similarity measure strongly wields influence on the number of 3D-2D correspondences. If it is set to a low value more map points are identified even so the amount of false matches is increased. By contrast, a high threshold reduces the number of matched 3D points but wrong correspondences are restricted as well.

Thus, the sequence has been tested with a variety of different thresholds. It turned out, that a value for the similarity measure beneath 0.7 is unsuitable for the sequence since the amount of false positives violates the assumption of approximately 5% outliers in the dataset. As a consequence, not all mismatches can be determined by the Tukey M-estimator and the optimized camera pose is badly influenced. By contrast, a threshold above 0.95 almost entirely discards spurious matchings. However, due to small perspective and illumination changes a huge amount of map points is not considered as found. This results in a less robust system and tracking usually fails before the next keyframe is acquired.

Therefore, the threshold for the normalized cross correlation is set to a value between 0.85 and 0.9. This provides almost continuously correct matches. The remaining false ones are easily identified by the Tukey M-estimator during the subsequent pose optimization. They are then assigned zero weight, effectively preventing them of contributing to the optimization process.

Next, the behaviour of the system under (large) scale changes is tested. Since the sequence starts with an almost pure forward motion it is ideal for this kind of evaluation. As a restriction, only the initial map is used for tracking and keyframes are never added to the system.

If the 3D-2D correspondence search is only performed in the pyramid level in which a map point was first observed, the approach is able to keep track of the camera for the first 110 frames. This corresponds to a distance of approximately 1.4m in z-direction. By contrast, if map points are searched in different pyramid levels, tracking first fails after a distance of about 2m. This behaviour indicates that tracking benefits from the usage of the pyramid-based approach.

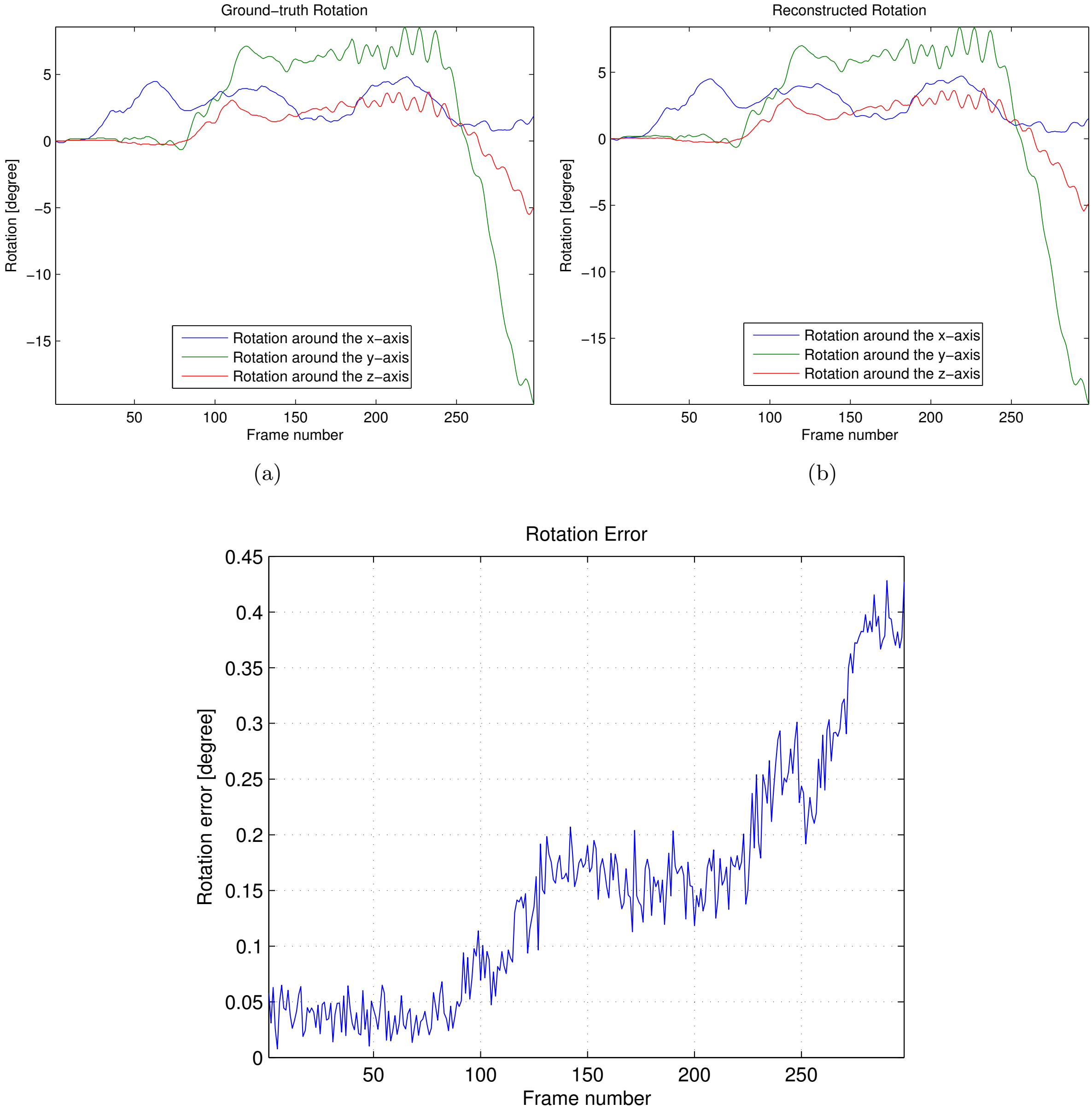
However, it is not clear which threshold should be used for an optimal and smooth transition between scale-spaces. The chosen values are empirical ones and no attempt has been made to derive a mathematical relation.

During evaluation, the distance between keyframes manifested as the parameter with the greatest impact on the results. On the one hand, if keyframes are set too sparse either tracking fails since the observed area is not sufficiently mapped or, at the most, new points cannot be added to the map. This is mainly due to the fact that the camera has traveled too far from the last keyframe and point correspondences cannot be established anymore. On the other hand, if keyframes are dropped too often the computation time is highly increased.

Another problem arises from the dominant forward motion of the camera: Since the correspondence search is only performed in an equal pyramid level, it is mandatory to drop a new keyframe even if the camera hardly has moved sideways. As stated in chapter 3.4.2 this yields a poor reconstruction and tracking has to continue with an uncertain map.

The sequence has been tested with a variety of different keyframe distances. Unfortunately, the approach is strongly liable to this parameter and could only handle a maximum distance of approximately 25cm in the x- and y-direction and about 30cm in the z-direction. If the distance exceeds this threshold adding new map points fails, whereas with a closer spacing computation time quickly gets out of hand. Therefore, an evaluation of different keyframe distances is omitted.

It should be noted that the patch-based approach has originally been developed for tracking a camera with dominant lateral movements. Correspondences between keyframes are therefore only searched in equal pyramid levels. Performing the search in different levels may reduce the sensitivity to the spacing between keyframes. This was not attempted in this thesis and still, forward motion in particular would remain challenging for the monocular SLAM approach.



(a)

(b)

(c)

Figure 7.5: Rotation accuracy of the patch-based approach for the New College dataset. (a) Ground-truth rotation. (b) Estimated rotation. (c) Rotation error.

The last part of the evaluation is concerned with the accuracy of the proposed method by comparing the estimated rotation and translation with the provided ground-truth.

Figure 7.5a shows the rotation for the first 300 frames of the sequence as given by the ground-truth, whereas figure 7.5b depicts the estimated rotation. The rotation error is illustrated in figure 7.5c.

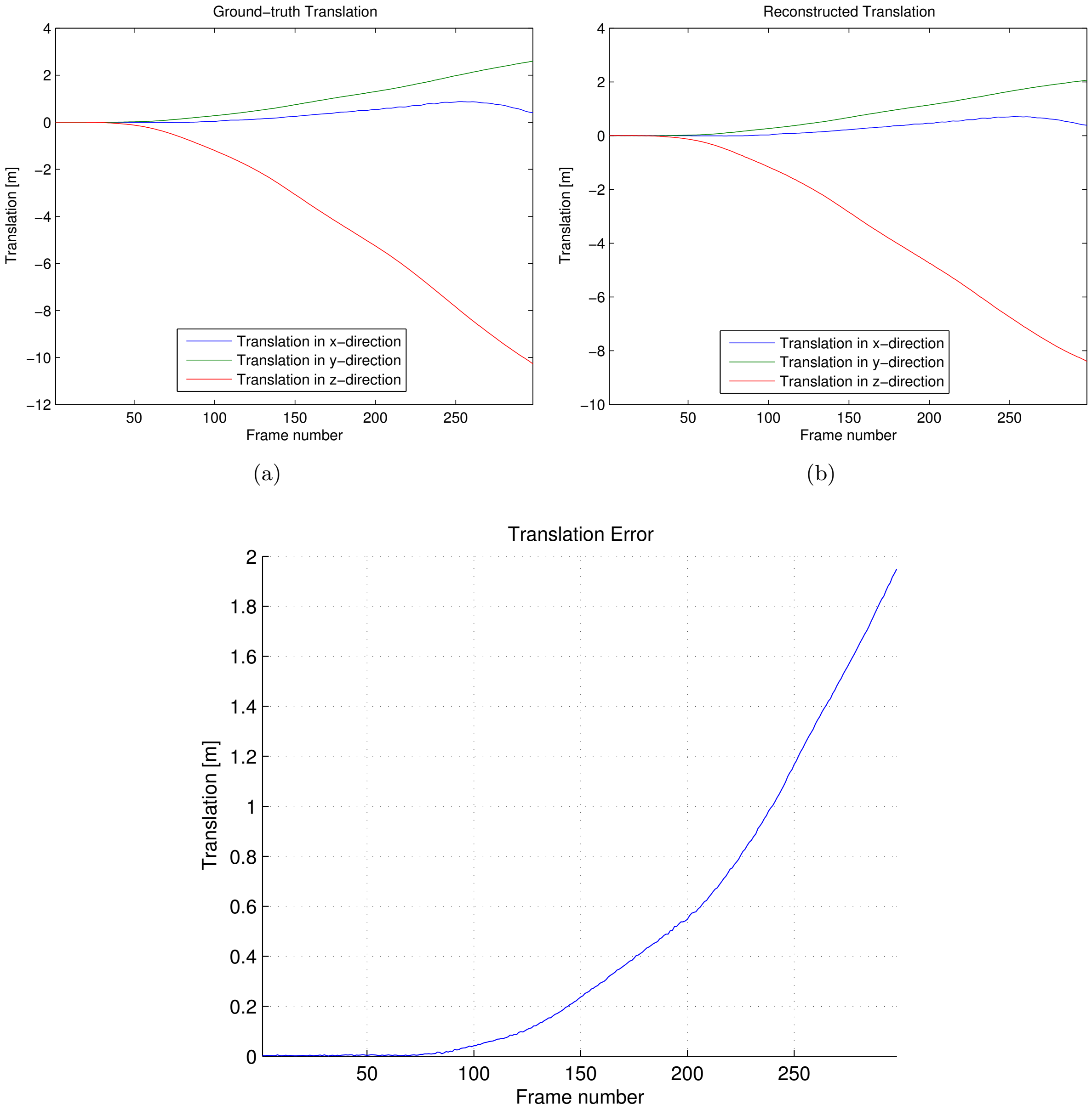
At first the camera remains stationary and there is almost no rotation. The tracking system can easily detect this and has a very small rotation error of about 0.05°. Around the 25th frame a small rotation around the x-axis is registered but this does not effect the accuracy of the estimate. However, as soon as the camera starts rotating around all the three axes the rotation error is slightly increased. This can be verified between the 80th and the 120th frame.

Subsequently, the rotation remains almost static and so does the rotation error. After the 250th frame the camera strongly rotates around the y-axis and additionally around the z-axis. This yields a further increase of the rotation error.

Nevertheless, the system performed obviously well with a final rotation error of approximately 0.4°.

By contrast, the approach exhibits serious problems with the estimation of the translation as it is shown in figure 7.6. Here, figure 7.6a illustrates the ground truth trajectory and figure 7.6b the estimated one. The trajectory is split into a translation in z-direction (red), y-direction (green) and x-direction (blue). The absolute trajectory error is shown in figure 7.6c.

The stationary stage of the camera is easily detected by the system and the translation error remains very small. After the 50th frame the robot starts to primarily move in the z-direction and shortly afterwards an additional motion in the y-direction is registered. At first this does not affect the system and the translation error is beneath 1cm. However, after the 80th frame the accuracy continually decreases until the end of the trajectory. It should be noted that the increase of the translation error after the 80th frame simultaneously comes along with a degradation of the rotation error as illustrated in figure 7.5c. This indicates that both estimates are closely related.

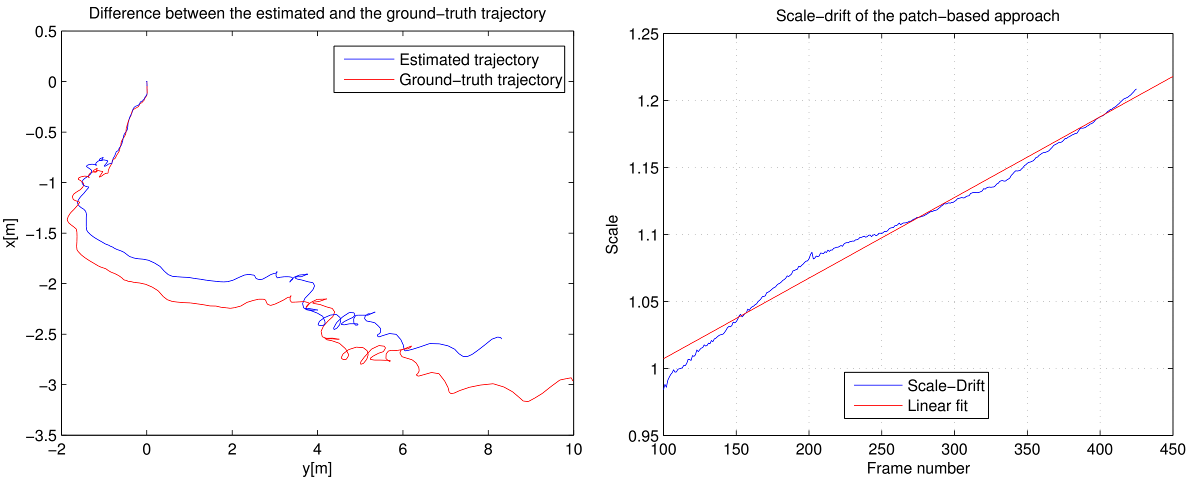


(a) (b)

(c)

Figure 7.6: Translation accuracy of the patch-based approach for the New College dataset. (a) Ground-truth translation. (b) Estimated translation. (c) Translation error.

A comparison between the ground truth trajectory (red) and the estimated trajectory (blue) for the first 425 frames is shown in figure 7.7a. Obviously, the estimated trajectory is essentially the same as the ground-truth trajectory but significantly shortened. Since the overall scale of the map cannot be determined in monocular SLAM, the estimated trajectory is aligned to the ground-truth trajectory by using the baseline between the stereo rig. If the system does not exhibit a scale-drift, the applied scale would remain constant throughout the entire sequence.



(a) (b)

Figure 7.7: (a) Comparison between the estimated and the ground-truth trajectory for the New College dataset using the patch-based approach. (b) Scale-drift of the patch-based approach for the New College dataset (blue). Linear fit (red).

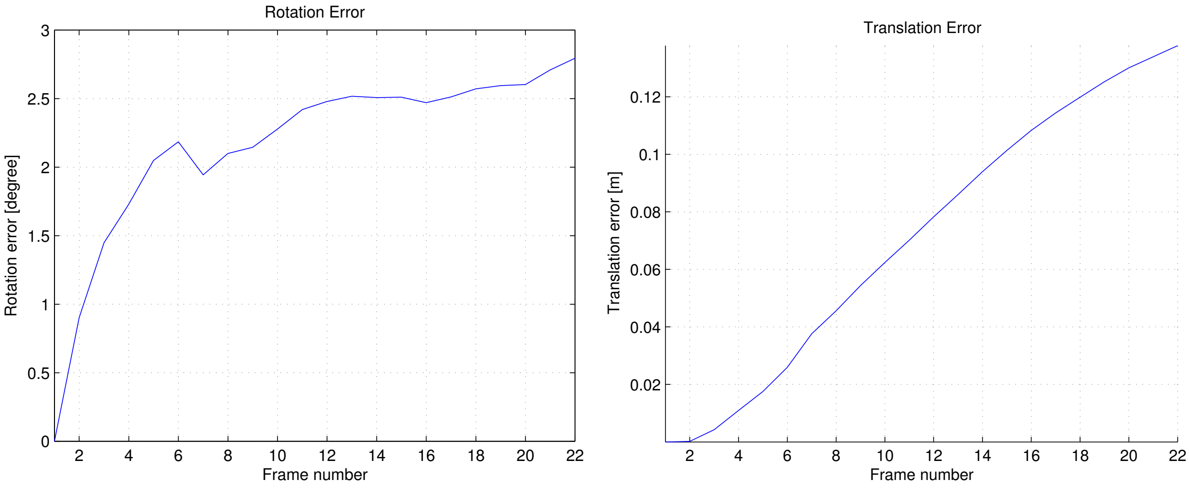
The blue curve in figure 7.7b shows the development of the scale factor between the 100th and the 425th frame, whereas the red one illustrates the linear fit. It is evident that the suggested approach suffers from a scale-drift which is responsible for the unsatisfying translation estimate. This behaviour is a well-known problem in monocular SLAM and it is system inherent. In the proposed method only a six degree of freedom (DOF) rigid transformation is considered. However, since the overall scale of the map cannot be determined there is an additional degree of gauge freedom. Strasdat et al. [52] showed that it is indeed the number of gauge degrees of freedom in which drift inevitably occurs. However, the drift only appears in large-scale SLAM and it is assumed that in particular the forward motion of the sequence contributes to the scale-drift. Therefore, the plan of using only the images of a single camera has been rejected. The map is initialized as always but the images of the stereo rig are processed alternating. For example, at frame three the image of the right camera is used, whereas at the subsequent frame the image of the left camera is processed. Even with this modification an improvement could not been determined and the cause of the scale-drift remains unclear. As it will be shown later, the SIFT approach does not exhibit this large drift.

Evaluation of the temple dataset

Next, the "Temple" sequence is evaluated. The main focus is to derive a statement of the system's ability to create an accurate map. In addition, the estimated trajectory is compared to the available ground-truth as well.

Since the temple is well textured, a vast amount of FAST corners can be detected in each image. Just like in the previous dataset, the threshold for patch-based matching is set to a value between 0.85 and 0.9. This discards most of the outliers but guarantees sufficient 3D-2D correspondences for robust tracking.

In comparison to the New College dataset the rotation and translation between consecutive frames is very large and the system has serious difficulties with the temple dataset. Although the system is able to track the camera for a few frames, this is not the case for the mapping thread. If the map is not extended after every frame, correspondence search fails and no map points can be added to the system. This degrades the proposed SLAM approach into some kind of structure from motion system. Nevertheless, the sequence has been evaluated in terms of the attainable accuracy as well. The result for the rotation error is shown in figure 7.8a, whereas the translation error is depicted in figure 7.8b.



(a) (b)

Figure 7.8: Translation and rotation accuracy of the patch-based approach for the temple dataset. (a) Rotation error. (b) Translation error.

Due to the large rotation between consecutive images, the approach has serious difficulties to obtain an accurate rotation estimate and the error increases to over 2° after only a few frames and finally to 2.75° at the end of the sequence. Likewise, the suggested approach has difficulties to estimate the translation. The error continually increases and finally differs 0.14m from the ground-truth.

A comparison between the ground-truth trajectory and the reconstructed trajectory is shown in figure 7.9.

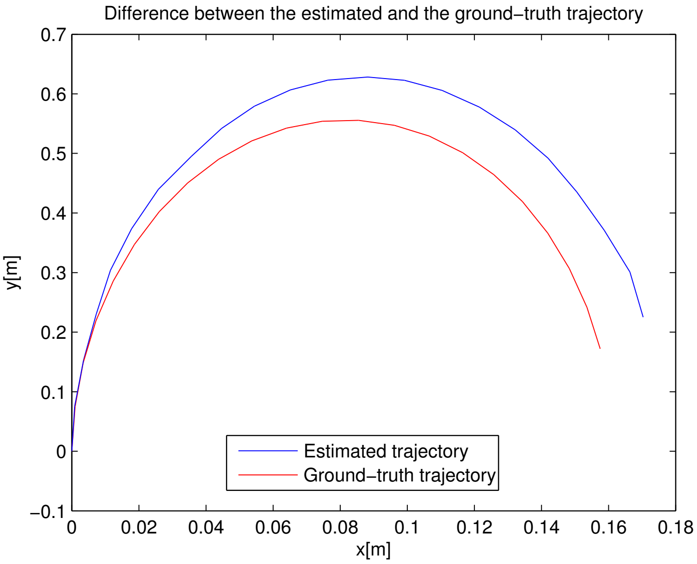
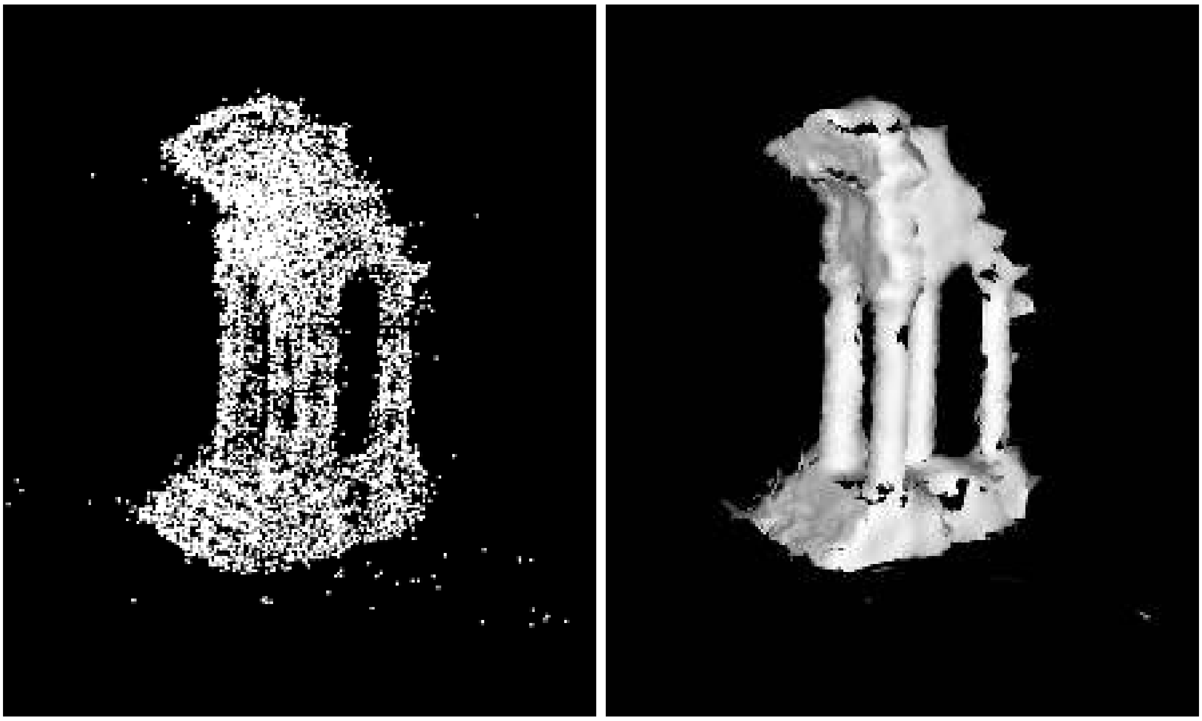


Figure 7.9: Trajectory of the patch-based approach for the Temple dataset

The resulting point cloud of the patch-based approach is illustrated in figure 7.10a. For the reconstruction, 22 frames where used yielding a map of about 12000 points. The structure of the temple is clearly visible and thus, the point cloud has been triangulated as well. The result is shown in figure 7.10b. The 3D object model exhibits several holes and the result is very smooth. This is mainly due to the fact that keypoints cannot be found in certain areas of the temple and that the overall point density is still too sparse. Furthermore, only a small subset of the entire ring has been used and the lack of map points in unvisited areas make a satisfying result impossible. However, the suggested approach is not designed for multi-view reconstruction. The vast amount of corners is simply for tracking robustness.



(a) (b)

Figure 7.10: Reconstruction of the temple using the patch-based approach. (a) Resulting point cloud. (b) Triangulated model.

Evaluation of the office dataset

Since the proposed method has originally been designed for tracking a camera in an area of limited size, the patch-based approach performed quite well in the office dataset. Although a huge part of some images contains self-similar texture, the system is able to detect and correctly match sufficient points needed for robust tracking.

The resulting trajectory is shown in figure 7.11. For reasons of clarity, only every tenth camera pose is used to illustrate the trajectory. Since no ground-truth is available, a quantitative statement cannot be derived. However, the camera started and ended at the exactly same point and the Euclidean distance between both points can be measured in order to get a feedback of the system's accuracy. After the metric scaling, the distance between those points was about 5mm which indicates a very good accuracy. It should be noted that no attempt has been made to detect and correct the loop.

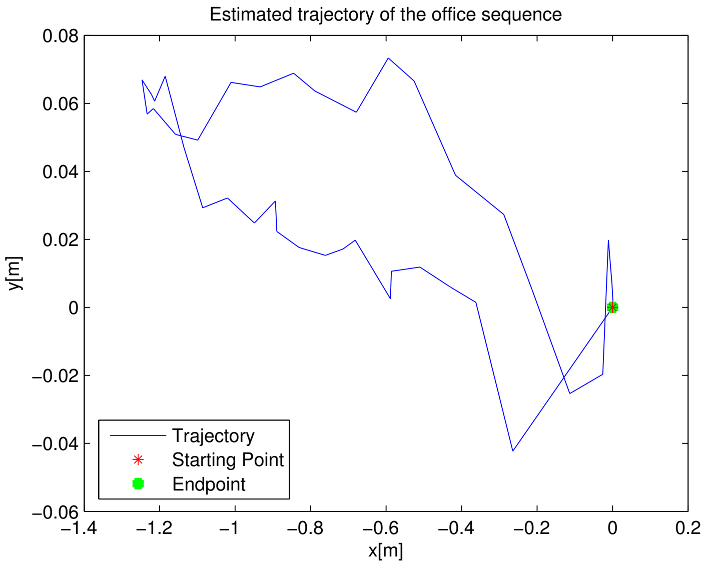


Figure 7.11: Trajectory of the office dataset using the patch-based approach. Note that the starting point overlaps the endpoint.

7.2.2 Evaluation Using SIFT-Features

Evaluation of the New College dataset

The same part of the New College sequence has been evaluated with the SIFT approach as well. Consequently, identical experiments have been run in order to provide a subsequent comparison between both approaches.

Since the SIFT features are even more discriminable than the patches the system performed very well and tracking never failed. In each part of the sequence sufficient map points were available or could been triangulated. Depending on the distance between keyframes and the nearest neighbour threshold the sequence contains between 3400 and 5000 map points for the first 300 frames and between 6500 and 12000 map points for the first 650 frames. Likewise dependent on the nearest neighbour threshold is the amount of identified map points in each frame. If a low value is used, up to 200 map points were on average identified. However, the good quality of the descriptors usually prevented false positives and remaining ones were easily identified during the pose optimization. By contrast, a high threshold decreases the number of successful observations to approximately 30 per frame.

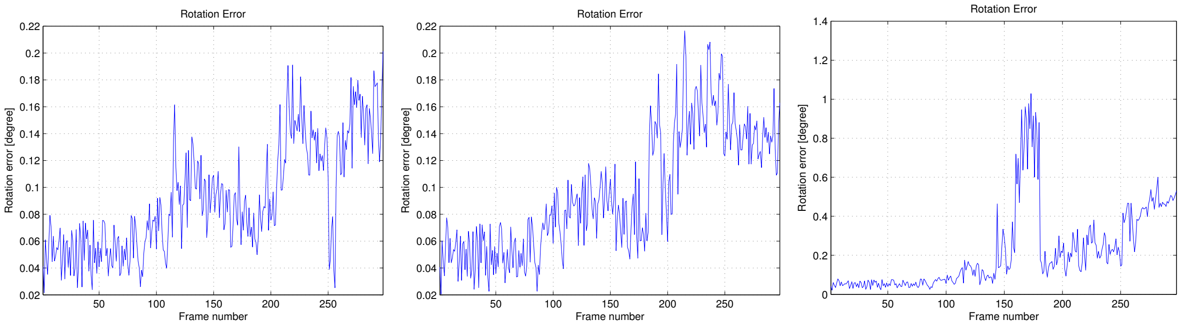
The sequence has been tested with a variety of different nearest neighbour thresholds but on no account a significant influence could be noticed. This indicates that the approach is able to obtain accurate pose estimates even if only few map points can be used for tracking. For further evaluations the threshold is set to a constant value such that approximately 100 map points can be identified in each image.

Just like the patch-based approach, the SIFT method has been evaluated in terms of the ability to handle (large) scale changes.

Since the detected keypoints and features are already scale-invariant neither an image pyramid nor an additional criterion in which level a specific map point has to be searched is necessary. The SIFT algorithm already examines a scale-space and finds corresponding features. Hence, the tracking system has no difficulties to estimate the camera pose despite the huge forward motion. If only the initial map is used, the system is able to track the camera for about 6m in z-direction.

However, not only tracking benefits of the scale-invariance but also the mapping thread: If correspondences between keyframes are found despite large scale changes, the distance between them can be set to a much larger value.

The advantage of this is two-fold: On the one hand, the computation time is reduced because bundle adjustment has to optimize less points and camera poses which enables a longer exploration in an acceptable time. On the other hand, the reconstructed map points are less uncertain since a sufficient baseline between consecutive keyframes is guaranteed.



(a) (b) (c)

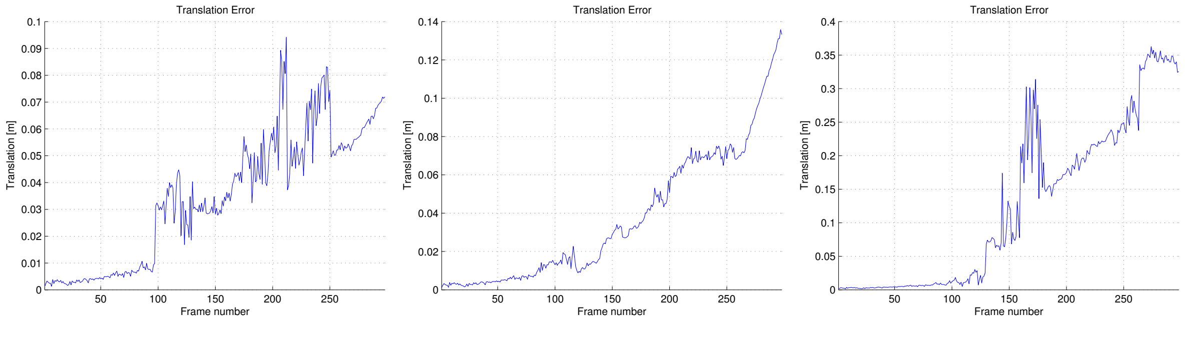
Figure 7.12: Rotation Error for Different Keyframe Distances

The sequence has been evaluated with several different keyframe distances. Three examples for the first 300 frames are shown in figure 7.12: On the left, the distance between keyframes is approximately 50cm in the x- and y-direction and about 75cm in the z-direction. Figure 7.12b shows the result with the same settings but the keyframes exhibit a distance of approximately 60cm in the x- and y-direction and about 100cm in the z-direction. The last result is shown in figure 7.12c where the spacing is 140cm in the x- and y-direction and about 180cm in the z-direction.

It can be seen that small distances between keyframes produce highly accurate rotation estimates with a small error of about 0.15° at the end of the sequence. However, if the distance is set to a larger value, the accuracy is highly decreased.

This behaviour reflects in the absolute translation error as well. Figure 7.13 shows the result for the same settings described above. It is obvious that the best result is obtained with the closest spacing. With an absolute trajectory error of approximately 14cm the moderate distance performs also well.

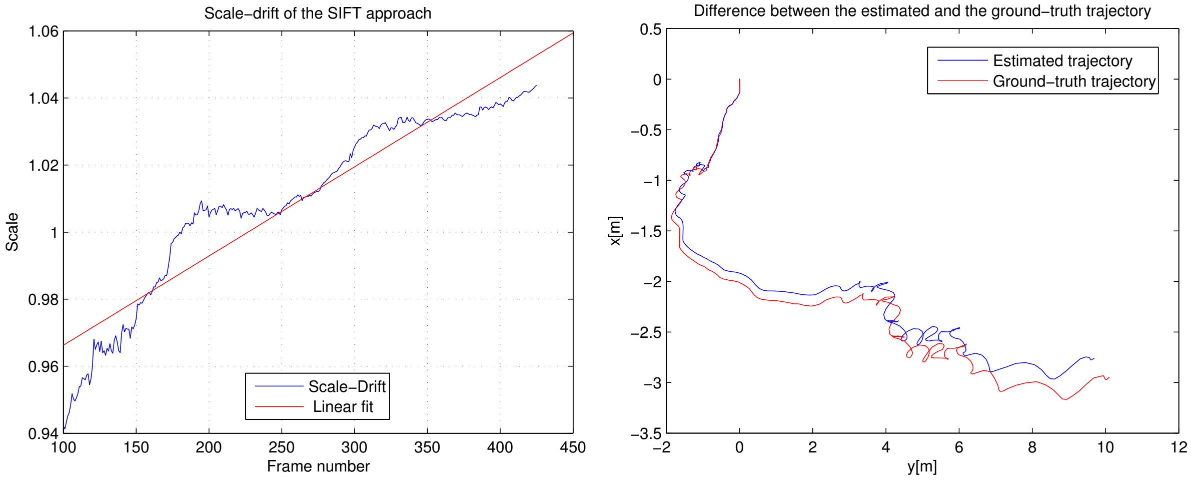
However, if the keyframes are spaced far away, the translation accuracy is degraded to 0.35cm within the same trajectory length. This indicates, that a reasonable choice of the keyframes is essential in order to obtain acceptable results.



(a) (b) (c)

Figure 7.13: Translation Error for Different Keyframe Distances

Beside the obtained accuracy, the scale-drift of the SIFT approach has been evaluated. As illustrated in figure 7.14a, the drift is greatly reduced compared with the patch-based approach. Since both methods exhibit the same system structure this behaviour is surprising. One explanation could be scale-invariance of the SIFT method. Since points from the initial map can be identified much longer, the coherence between map fragments is greater and the drift may be reduced. However, this cannot be proven and it remains ambiguous if this is the true reason.



(a) (b)

Figure 7.14: (a) Scale-drift of the New College dataset using the SIFT approach. (b) Trajectory of the New College dataset using the SIFT approach.

For completeness, the resulting trajectory for the first 425 frames is shown in figure 7.14b. As expected from the improved accuracy, the trajectory estimate is better than in the patch-based approach. However, due to the small scale-drift, the trajectory is also slightly shorter.

Evaluation of the temple dataset

The SIFT approach has also been evaluated on the temple dataset.

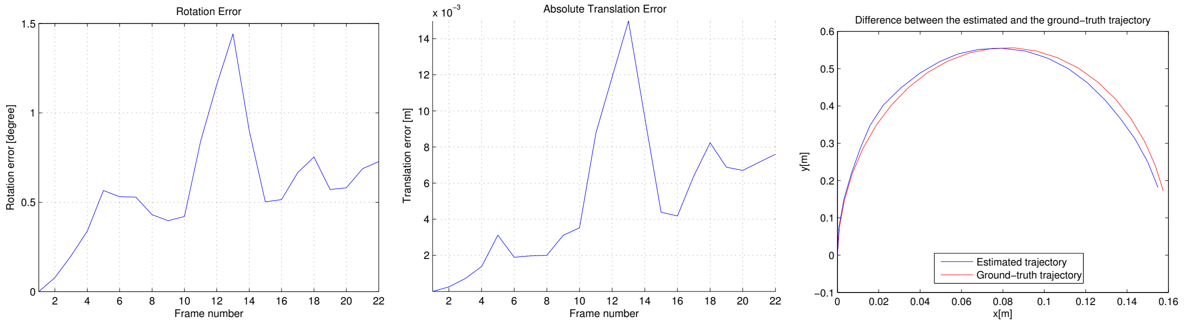
Since the temple exhibits sufficient texture, a few hundreds of keypoints can be detected in each frame. Due to the strongly discriminable SIFT features a huge amount of these are then correctly matched. This clearly depends on the preset threshold for the nearest neighbour but like in the New College dataset, the SIFT approach can handle a wide range of different settings without compromising the accuracy.

Despite the large motion between consecutive frames the SIFT approach is easily able to track the camera for a few frames without extending the map. However, an accurate map is only obtained if a new keyframe is acquired after each frame.

Figure 7.15a shows the accuracy of the estimated rotation. The accuracy quickly decreases after a few frames and remains then relatively constant. The final rotation error is less than 0.8° compared to ground-truth.

In figure 7.15b the obtained translation accuracy is shown. The graph exhibits the same structure as the rotation estimate indicating, that both are highly correlated.

In figure 7.15c the ground-truth trajectory (red) and the estimated trajectory is shown. Obviously, the SIFT approach performed very well in terms of accuracy.



(a) Absolute rotation error (b) Absolute translation error (c)

Figure 7.15: Absolute rotation and translation error for the temple dataset using the SIFT approach

Another part of the evaluation was to test the mapping abilities of the SIFT approach. Since the system only maintains a few but highly discriminable features, the produced maps are very sparse. Even if almost all detected points are stored, the map contains only approximately 4000 points. Compared to the patch-based approach, which constructed a map of about 12000 points, the result is unsatisfying and an illustration is therefore omitted.

Evaluation of the office dataset

Finally, the performance of the SIFT approach is analysed by evaluating the office dataset. As in the other datasets, the system was able to detect and match sufficient points to provide accurate tracking.

The resulting trajectory is shown in figure 7.16. The Euclidean distance between the starting point and the endpoint is only 4mm and thus the SIFT approach performed very well in terms of the obtainable accuracy.

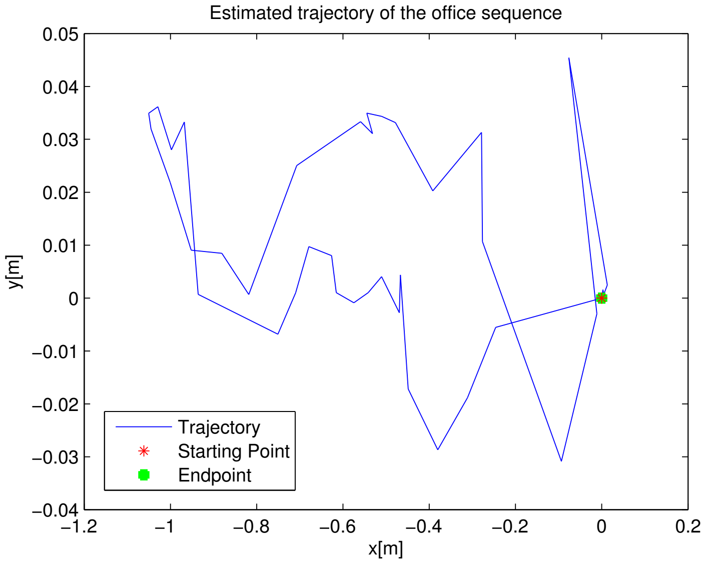


Figure 7.16: Trajectory of the office dataset using the SIFT approach. Note that the starting point overlaps the endpoint.

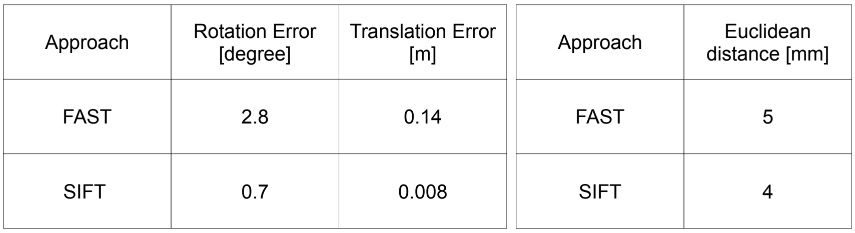
7.3 Comparison

Both suggested approaches were evaluated in terms of accuracy, scale-changes, scale-drift and mapping quality. It became obvious, that the SIFT approach outperformed the patch-based approach in all datasets with regards to accuracy and tracking robustness.

Furthermore, the system can handle scale changes more easily since the SIFT method already examines a scale-space. Especially the mapping benefits from the scale-invariance, because the distance between keyframes can be increased if the dominant motion of the camera is forwards. If the spacing is only increased up to a certain degree, the accuracy is not degraded but the computational time is significantly reduced.

Another advantage of the SIFT approach is the reduced scale-drift. This was particularly obvious in the New College dataset in which the camera traveled a long distance. The FAST approach was not able to provide an accurate estimate of the translation, whereas the accuracy of SIFT was in the range of a few decimetres. The reason of the increased scale-drift of the patch-based approach could not be found, but it is assumed that due to the better coherence of the SIFT approach the scale-drift is reduced.

The only part in which the patch-based approach provides better results is in terms of the mapping quality. Since a vast amount of keypoints is detected in four pyramid levels, the map contains significantly more points than the one created by the SIFT approach.
An overview of the obtained accuracy for the datasets is shown in the subsequent figures. In figure 7.17a the rotation and translation error of the temple dataset is shown. Obviously, the SIFT approach outperformed the patch-based method. In contrast, both approaches exhibit almost the same accuracy for the office dataset (see figure 7.17b). It should be noted that the patch-based approach has been developed for exactly this kind of sequence.



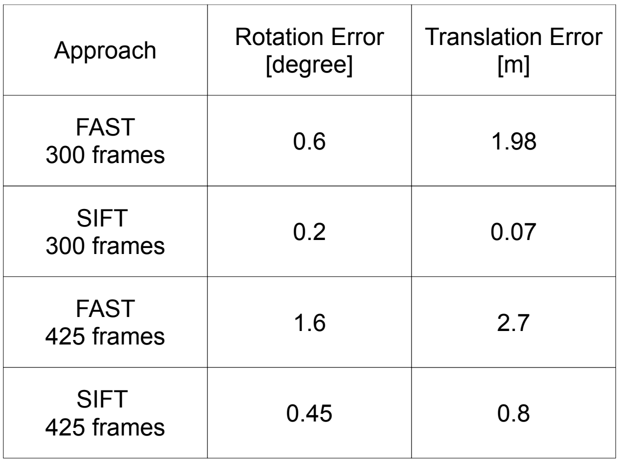
Approach	Rotation Error [degree]	Translation Error [m]
FAST	2.8	0.14
SIFT	0.7	0.008

Approach	Euclidean distance [mm]
FAST	5
SIFT	4

(a) (b)

Figure 7.17: Comparison of the accuracy for the temple and the office sequence. (a) Accuracy for the temple dataset. (b) Accuracy for the office sequence.

In figure 7.18 the rotation and translation accuracy for the New College dataset is shown. It is evident that the estimates of the SIFT approaches are more accurate. Especially if only the first 300 frames are considered, the system exhibits an exceptional translation and rotation accuracy. In summary it can be stated that the SIFT approach is superior in terms of tracking performance and accuracy, whereas the FAST approach creates a map that is better suited for further usage.



Approach	Rotation Error [degree]	Translation Error [m]
FAST 300 frames	0.6	1.98
SIFT 300 frames	0.2	0.07
FAST 425 frames	1.6	2.7
SIFT 425 frames	0.45	0.8

Figure 7.18: Comparison of the accuracy for the New College dataset

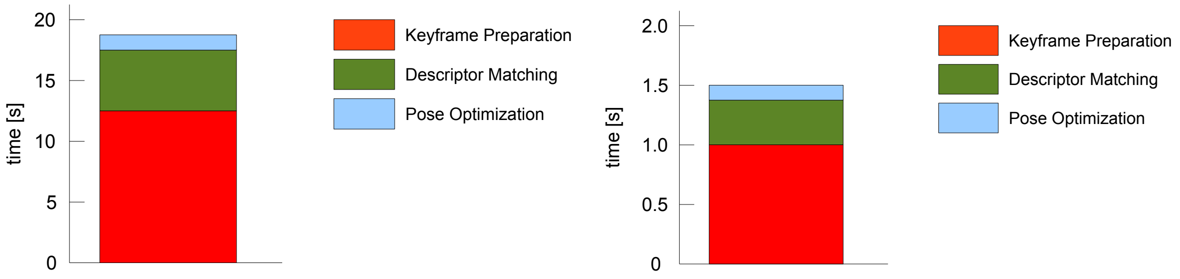
7.4 Processing time

Both approaches have been implemented and evaluated in Matlab. In the following, both methods are examined concerning their overall runtimes. The runtime can be split into two different parts:

- Local localization
- Map extension

Local Localization

The local localization can be divided into three main parts: Keyframe preparation, descriptor matching and pose optimization. The keyframe preparation includes keypoint detection and creating the image pyramid (if necessary). The descriptor matching consists of projecting map points, determining the potentially visible set and establishing the 3D-2D correspondences. This step is clearly dependent on the map size and the number of points that are currently visible. The patch-based approach employs the FAST corner algorithm for keypoint detection. This choice has been made, because the timing results obtained by Klein and Murray [22] promised a real-time capability. However, the Matlab implementation of the detector is very slow. If performed on a commodity system consisting of an i7 quad-core CPU and four GB RAM, the extraction of FAST corners on images with a resolution of 640x480 takes approximately 12 seconds. Figure 7.19a shows the tracking timings for a small map consisting of approximately 1000 map points. It is evident that most of the time is spent on keypoint detection. The subsequent descriptor matching needs only about a third of this time and the pose optimisation is negligible compared with the rest.



(a) (b)

Figure 7.19: Required time for local localisation. (a) Timings for the patch-based approach. (b) Timings for the SIFT approach.

Figure 7.19b shows the timings required by the SIFT approach. It is somewhat surprising that SIFT outperforms the patch-based approach in terms of processing time. This is mainly due to the usage of a mex-file which provides the inclusion of subroutines written for example in C. Therefore, the keyframe preparation as well as the descriptor matching can be done in less than one second. The subsequent pose refinement is also slightly faster because the usually less 3D-2D correspondences are found compared with the patch-based approach. All things considered, the processing time for a frame using the SIFT approach is on average 1.5 seconds.

Map extension

The map extension can be divided into two main parts: Adding new map points and optimizing structure and motion. Unfortunately, the mapping thread scales poorly with increasing map size and the employed bundle adjustment can be considered as the bottleneck of the suggested approach. Even if the map consists only of a few hundreds of points and about five poses bundle adjustment needs seconds to converge. This time drastically increases when new points and poses are added to the map. Therefore, the remaining operations such as correspondence search or the triangulation of new map points are negligible.

Beside the Matlab implementation the construction of the initial map for the FAST corner approach has been written in C as well. Thereby, the computation time could be significantly reduced: Even with non-maximum suppression, the algorithm was able to initialize the map in less than 40ms while only 4ms were required for keypoint detection. This indicates, that the approach might be real-time capable if fully written in C.

7.5 Discussion

In this chapter the two proposed approaches were evaluated in terms of their performance on three different datasets. The first approach is based on FAST corners and patch-based matching. It intends to obtain a robust estimate of the current camera pose by identifying a vast amount of map points that are represented by weak descriptors. During evaluation it became apparent that the approach is prone to the threshold used for patch-based matching. If it is set too low, the 3D-2D correspondences contain many outliers and the optimization result is badly influenced. By contrast, if it is set too high, the robustness of the system is decreased.

Despite these drawbacks, the approach was able to track the camera throughout the entire New College sequence. The obtained rotation accuracy was acceptable, whereas the translation accuracy lagged behind. A closer look revealed that the system exhibits a large scale-drift which is a common problem in monocular SLAM. However, since the camera traveled only a small distance, scale-drift should not be caused in this extend. The reason for this behaviour could not be found.

The approach was also able to keep track of the camera in the temple dataset. However, the huge motion between consecutive frames caused some problems and the accuracy was worse in comparison with the SIFT approach. By contrast, the reconstructed model benefited from the vast amount of map points and the result was quite satisfying.

Since the approach was originally developed for hand-held SLAM in small environments, it could also handle the third dataset and the accuracy was almost as good as the one obtained by the SIFT approach.

The second approach is based on DoG keypoints and strong feature descriptors. It intends to estimate the current camera pose with a few but highly distinctive features. In the experiments, the advantage of this appeared in several ways. First, the approach is not prone to the feature matching threshold and a robust pose estimate is obtained with a variety of different values. This is a great improvement compared to the patch-based approach, in which a suitable threshold first wants to be found. Furthermore, the usage of scale-invariant feature performed significantly better than the employed pyramid level of the patch-based approach.

However, the analysis of the processing time revealed some bottlenecks: Feature detection and bundle adjustment. The processing time for the patch-based approach already has been reduced to an acceptable amount by using a C implementation. Since the SIFT algorithm is known for its high processing time, the only possibility is to use another approach. One example for a promising feature detector is Speeded Up Robust Features (SURF) [4]. However, it remains to be seen, if this yields a robust and real-time capable SLAM system.

Speeding up bundle adjustment is more difficult but there exist promising methods such as the approach of Sibley et al. [43] as it was explained in chapter 2.

CHAPTER 8

Conclusion and Future work

This thesis is concerned with the problem of simultaneously estimating the camera pose while creating a three dimensional map of the observed environment. Therefore, two different approaches have been developed. Both are strongly related to the seminal work of Klein and Murray [22] that split tracking and mapping into two different subtasks executed in separated threads. Both approaches take up this proposal and exhibit almost the same system structure. This implies not only the usage of keyframes that are carefully chosen snapshots of the incoming video stream but also the detection of interest points and feature-based matching.

However, both approaches use different keypoint detectors and matching strategies. On the one hand, the FAST corner detector in combination with patch-based feature matching is employed. On the other hand, DoG keypoints are extracted and represented via the strongly discriminable SIFT features.

Beside these changes the procedure is exactly the same in both cases: At first an initial map is constructed using a stereo algorithm. Subsequently, the pose of the camera is estimated at each incoming frame exploiting the built map. Whenever the camera leaves the mapped area, the system automatically extends the map by adding new 3D points.

One goal of this thesis was the robust estimation of the camera pose. Both approaches fulfil this requirement, albeit through a different procedure. The FAST corner approach gains its robustness due to the huge amount of points that can be detected in each frame. The additional usage of an image pyramid further increases the number of points and enables tracking despite scale-changes. However, the approach exhibits some drawbacks that appeared during evaluation. Due to the weak discriminable patches that are used for correspondence search, the approach is prone to mismatches.

By contrast, the SIFT approach starts from the premise that less but more discriminable points are sufficient for robust tracking. In addition, the detected keypoints and their corresponding descriptors are already scale-invariant and no image pyramid is necessary.

Another goal of this thesis was the accurate reconstruction of an observed scene. Therefore, the mapping thread employs the keyframe concept. A keyframe is a snapshot that is occasionally extracted from the incoming video stream. Only the keyframes contribute to the information that is contained in the map. Since the mapping thread has to deal with inaccurate camera poses, the 3D points cannot be optimally reconstructed. Therefore, a joint optimization over all map points and camera poses is performed. This is done by the computationally demanding but highly accurate concept of bundle adjustment.

In chapter 7 the proposed methods were evaluated on several different datasets. At first, a typical SLAM dataset was examined and it was shown, that both approaches were able to track the camera through the entire sequence. However, the patch-based method is prone to the distance between keyframes and in addition it exhibits a large scale-drift making an accurate estimation of the translation impossible. By contrast, the SIFT approach attended not only with greatly increased accuracy and reduced scale-drift but also with a fast implementation due to the inclusion of a mex-file.
Subsequently, both methods were evaluated in terms of their ability to reconstruct an observed scene using a multi-view stereo reconstruction dataset. Although the feature-based approach outperformed the patch-based in terms of accuracy, the latter was able to reconstruct a better usable map.
Finally, both approaches were evaluated on a self-recorded sequence. Both methods had an excellent accuracy within a few millimetres.

All things considered, especially the SIFT approach performed well in all evaluated datasets and thus, a reliable base for future research has been created. In the following, research challenges together with possible solutions are presented and various possibilities to improve the accuracy, robustness and processing time of the system are outlined as well.

So far, the proposed method is entirely written in Matlab and not nearly real-time capable. This is a huge drawback since neither a robot nor a user can wait tens of seconds to continue. Therefore, a C implementation has been started. The first evaluations in terms of processing time are very promising and indicate, that real-time is reachable.
However, even with a well written C code, the cubic complexity of bundle adjustment would soon prevent real-time exploration. One common possibility to overcome this problem is the employment of a sliding window that optimizes only over a subset of all poses and map points. Another possibility is the usage of relative bundle adjustment as proposed by [43].
One main problem of the proposed method is motion blur. This does not only effect the local localization but also the map building process and can yield a system failure. However, there exist promising approaches that allow accurate camera tracking despite the presence of motion

blur. Klein and Murray [22] demonstrated that the adding of small lines, the so-called edgelets, greatly improves tracking robustness even if the images are blurred. Since the entire system remains the same this would only need a small intervention.
Another approach that holds a lot of promise is the work of Newcombe et al. [36]. They showed that it is possible to obtain a dense reconstruction of the observed environment in real-time while simultaneously using the recently created model for camera tracking. The resulting approach exhibits superior tracking performance compared with feature-based systems.
Another main drawback of the proposed method and all vision based approaches in particular is the fact, that they cannot handle repetitive textures such as equally painted walls. One possibility to overcome not only this problem but also to improve the accuracy and robustness of vision based approaches is sensor fusion. In situations in which all sensors work, the pose estimate can be refined combining all measurements, whereas in situations in which visual odometry is not possible the additional sensors prevent the system from failing.
There exists a bunch of suitable sensors such as global navigate satellite system (GNSS) receivers or inertial measurement units (IMU). Both can support the pose estimate with either absolute measurements in the case of GNSS or accurate relative information in terms of rotations and accelerations in the case of the latter.
Furthermore, one essential part of SLAM has not been considered throughout this thesis: Loop closures. Since the proposed approach is an incremental method, small errors sum up resulting in a large error at the end of the trajectory. The detection and correction of a loop is an important part of SLAM systems and should clearly be added to the existing approaches.

Revisiting the goals of chapter 1 the proposed approaches are indeed able to robustly track a camera in a previously unknown area while simultaneously build a three dimensional map of the observed scene. It was shown in an experiment, that the methods is also able to accurately track a hand-held camera. However, the suggested method in its current implementation is by far not real-time capable but the first steps to reach this goal are taken.

List of Figures

1.1 Currently Existing Autonomous Robots and Self-Driving Cars 1

2.1 Different Representations of the SLAM Problem 6
2.2 Absolute and Relative Representations 11
2.3 Inverse Depth Map 13

3.1 Rigid Transformation 16
3.2 Angle-Axis Representation 16
3.3 Camera Model 18
3.4 Image Plane 19
3.5 Epipolar Geometry 24
3.6 Extraction of Camera Matrices from the Essential Matrix 29
3.7 Uncertainty of Reconstruction 30
3.8 A Typical Bundle Adjustment Problem 31
3.9 Form of the Jacobian Struture in Bundle Adjustment 35
3.10 Types of Different Cost Functions 39

4.1 Separation of Tracking and Mapping 41
4.2 Overview of the Proposed SLAM System 42

5.1 Local Localization 47

7.1 Aerial View of the New College Dataset 62
7.2 Snapshots of the New College Dataset 63
7.3 Snapshots of the Middelbury Dataset 64
7.4 Snapshots of the Office Dataset 65
7.5 Rotation Accuracy for the New College Dataset (patch-based) 67
7.6 Translation Accuracy for the New College Dataset (patch-based) 69
7.7 Trajectory of the New College Dataset (patch-based) 70

7.8 Accuracy for the Temple Dataset (patch-based) 71
7.9 Trajectory for the Temple Dataset (patch-based) 72
7.10 Reconstruction of the Temple (patch-based) 73
7.11 Trajectory of the Office Dataset (patch-based) 74
7.12 Rotation Error for Different Keyframe Distances 75
7.13 Translation Error for Different Keyframe Distances 76
7.14 Trajectory of the New College Dataset (SIFT) 76
7.15 Accuracy of the SIFT Approach on the Temple Dataset 77
7.16 Trajectory of the Office Dataset (SIFT) 78
7.17 Comparison of the Accuracy for the Temple and the Office Dataset 79
7.18 Comparison of the Accuracy for the New College Dataset 79
7.19 Processing Time Local Localisation 80

Bibliography

[1] Angle, C. M. (2011, February 10). iRobot CEO Discusses Q4 2010 Results - Earnings Call Transcript. [online], `seekingalpha.com/article/252090`. (Accessed 20 April 2013).

[2] Bailey, T., Nieto, J. and Nebot, E. (2006). Consistency of the FastSLAM algorithm, *in* 'Proceedings of the 2006 IEEE International Conference on Robotics and Automation', pp. 424–429.

[3] Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms, *in* 'Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition', Vol. 1, pp. 1090–1097.

[4] Bay, H., Tuytelaars, T. and Van Gool, L. (2006). Surf: Speeded up robust features, *in* 'Computer Vision–ECCV 2006', Springer, pp. 404–417.

[5] Betge-Brezetz, S., Hebert, P., Chatila, R. and Devy, M. (1996). Uncertain map making in natural environments, *in* 'Proceedings of the 1996 IEEE International Conference on Robotics and Automation', Vol. 2, pp. 1048–1053.

[6] Brown, D. C. (1958). A solution to the general problem of multiple station analytical stereo-triangulation, RCA-MTP data reduction, Technical report.

[7] Castellanos, J. A., Martinez-Cantin, R., Tardós, J. D. and Neira, J. (2007). Robocentric map joining: Improving the consistency of EKF-SLAM, *Robotics and Autonomous Systems* 55(1), 21–29.

[8] Castellanos, J. A., Neira, J. and Tardós, J. D. (2004). Limits to the consistency of EKF-based SLAM, *in* 'Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles', Lisbon, Portugal.

[9] Castle, R., Klein, G. and Murray, D. W. (2008). Video-rate localization in multiple maps for wearable augmented reality, *in* 'Proceedings of the 12th IEEE International Symposium on Wearable Computers', pp. 15–22.

[10] Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera, *in* 'Proceedings of the ninth IEEE International Conference on Computer Vision', pp. 1403–1410.

[11] Davison, A. J., Reid, I. D., Molton, N. D. and Stasse, O. (2007). MonoSLAM: real-time single camera SLAM, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(6), 1052–1067.

[12] Dellaert, F., Fox, D., Burgard, W. and Thrun, S. (1999). Monte carlo localization for mobile robots, *in* 'Proceedings of the 1999 IEEE International Conference on Robotics and Automation', Vol. 2, pp. 1322–1328.

[13] 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.

[14] Guizzo, E. (2011, Octobre 18). How Google's Self-Driving Car Works. [online], *http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works*. (Accessed 20 April 2013).

[15] Hahnel, D., Burgard, W., Fox, D. and Thrun, S. (2003). An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements, *in* 'Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems', Vol. 1, pp. 206–211.

[16] Harris, C. and Stephens, M. (1988). A combined corner and edge detector, *in* 'Proceedings of the 4th Alvey vision conference', Vol. 15, Manchester, UK, pp. 147–151.

[17] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*, second edn, Cambridge University Press.

[18] 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', pp. 2264–2269.

[19] Huber, P. J. (1981). *Robust Statistics*, John Wiley & Sons.

[20] Julier, S. J. and Uhlmann, J. K. (2001). A counter example to the theory of simultaneous localization and map building, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', Vol. 4, pp. 4238–4243 vol. 4.

[21] Jung, I. K. and Lacroix, S. (2003). High resolution terrain mapping using low attitude aerial stereo imagery, *in* 'Proceedings of the ninth IEEE International Conference on Computer Vision', pp. 946–951.

[22] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces, *in* 'Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality', pp. 225–234.

[23] Klein, G. and Murray, D. (2008). Improving the agility of keyframe-based SLAM, *in* 'Proceedings of the 2008 European Conference on Computer Vision', pp. 802–815.

[24] Klein, G. and Murray, D. (2009). Parallel tracking and mapping on a camera phone, *in* 'Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality', pp. 83–86.

[25] Lavoie, S. (2011, July 22). Pia14309: Mars science laboratory mission's curiosity rover (stereo). [online], *http://photojournal.jpl.nasa.gov/catalog/PIA14309.* (Accessed 20 April 2013).

[26] Lowe, D. G. (1999). Object recognition from local scale-invariant features, *in* 'Proceedings of the seventh IEEE International Conference on Computer vision', Vol. 2, pp. 1150–1157.

[27] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints, *International journal of computer vision* 60(2), 91–110.

[28] Mei, C., Sibley, G., Cummins, M., Newman, P. and Reid, I. (2011). RSLAM: a system for large-scale mapping in constant-time using stereo, *International journal of computer vision* 94(2), 198–214.

[29] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B. (2002). FastSLAM: a factored solution to the simultaneous localization and mapping problem, *in* 'Proceedings of the 2002 National conference on Artificial Intelligence', pp. 593–598.

[30] Moutarlier, P. and Chatila, R. (1990). Stochastic multisensory data fusion for mobile robot location and environmental modelling, *in* 'H. Miura, ed., Fifth International Symposium of Robotics Research', MIT Press, Cambridge, MA, USA, pp. 85–94.

[31] Murray, R. M., Li, Z., Sastry, S. S. and Sastry, S. S. (1994). *A mathematical introduction to robotic manipulation*, CRC PressI Llc.

[32] Murrieta-Cid, R., Parra, C. and Devy, M. (2002). Visual navigation in natural environments: from range and color data to a landmark-based model, *Autonomous Robots* 13(2), 143–168.

[33] Neira, J., Ribeiro, M. I. and Tardos, J. D. (1997). Mobile robot localization and map building using monocular vision, *in* 'Proceedings of the 5th Symposium for Intelligent Robotics Systems', pp. 275–284.

[34] 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.

[35] Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera, *in* 'Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition', pp. 1498–1505.

[36] Newcombe, R. A., Lovegrove, S. J. and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time, *in* 'Proceedings of the 2011 IEEE International Conference on Computer Vision', pp. 2320–2327.

[37] Nistér, D. (2004). An efficient solution to the five-point relative pose problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(6), 756–770.

[38] Paz, L., J., G., Tardós, J. D. and Neira, J. (2007). Data Association in O(n) for Divide and Conquer SLAM, *in* 'Proceedings of Robotics: Science and Systems', Atlanta, GA, USA.

[39] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection, *in* 'European Conference on Computer Vision', Vol. 1, pp. 430–443.

[40] Rousseeuw, P. J. and Leroy, A. M. (2005). *Robust regression and outlier detection*, Vol. 589, Wiley.

[41] Scharstein, D. (2012, April 18). Multi-view stereo evaluation web page. [online], *http://vision.middlebury.edu/mview/*. (Accessed 20 April 2013).

[42] Seitz, S. M., Curless, B., Diebel, J., Scharstein, D. and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms, *in* 'Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 1, pp. 519–528.

[43] Sibley, G., Mei, C., Reid, I. and Newman, P. (2009). Adaptive relative bundle adjustment, *in* 'Robotics Science and Systems Conference', pp. 1–8.

[44] Sim, R., Elinas, P., Griffin, M., Little, J. et al. (2005). Vision-based slam using the rao-blackwellised particle filter, *in* 'IJCAI Workshop on Reasoning with Uncertainty in Robotics', Vol. 14, pp. 9–16.

[45] Smith, M., Baldwin, I., Churchill, W., Paul, R. and Newman, P. (2009). The new college vision and laser data set, *The International Journal of Robotics Research* 28(5), 595–599.

[46] Smith, R., Self, M. and Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships, *in* 'Proceedings of the 4th international symposium on Robotics Research', MIT Press, Cambridge, MA, USA, pp. 467–474.

[47] Steger, C., Ulrich, M. and Wiedemann, C. (2008). *Machine Vision Algorithms and Applications*, Wiley-VCH.

[48] Stewénius, H., Engels, C. and Nistér, D. (2006). Recent developments on direct relative orientation, *ISPRS Journal of Photogrammetry and Remote Sensing* 60(4), 284–294.

[49] Stühmer, J., Gumhold, S. and Cremers, D. (2010). Real-time dense geometry from a handheld camera, *Pattern Recognition* pp. 11–20.

[50] Strasdat, H., Davison, A. J., Montiel, J. M. M. and Konolige, K. (2011). Double window optimisation for constant time visual SLAM, *in* 'Computer Vision (ICCV), 2011 IEEE International Conference on', IEEE, pp. 2352–2359.

[51] Strasdat, H., Montiel, J. M. M. and Davison, A. J. (2010*a*). Real-time monocular SLAM: Why filter?, *in* 'Proceedings of the 2010 IEEE International Conference on Robotics and Automation', pp. 2657–2664.

[52] Strasdat, H., Montiel, J. M. M. and Davison, A. J. (2010*b*). Scale drift-aware large scale monocular SLAM, *in* 'Proceedings of Robotics: Science and Systems (RSS)', Vol. 2, p. 5.

[53] Strasdat, H., Montiel, J. M. M. and Davison, A. J. (2012). Visual SLAM: why filter?, *Image and Vision Computing* .

[54] Tardós, J. D., Neira, J., Newman, P. M. and Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data, *The International Journal of Robotics Research* 21(4), 311–330.

[55] Thrun, S., Burgard, W. and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press.

[56] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M. and Hoffmann, G. (2006). Stanley: The robot that won the DARPA grand challenge, *Journal of field Robotics* 23(9), 661–692.

[57] Triggs, B., McLauchlan, P. F., Hartley, R. I. and Fitzgibbon, A. W. (2000). Bundle adjustment—a modern synthesis, *in* 'Vision algorithms: theory and practice', Springer, pp. 298–372.

[58] Wang, C.-C., Thorpe, C. and Thrun, S. (2003). Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas, *in* 'Proceedings of the 2003 IEEE International Conference on Robotics and Automation', Vol. 1, pp. 842–849.

[59] Wijk, O. and Christensen, H. I. (2000). Localization and navigation of a mobile robot using natural point landmarks extracted from sonar data, *Robotics and Autonomous Systems* 31(1), 31–42.

[60] Willow Garage (2008-2011). Hardware and Software Platform for Mobile Manipulation R&D. [online], `http://www.willowgarage.com/pr2`. (Accessed 20 April 2013).

[61] Zhang, Z. (2000). A flexible new technique for camera calibration, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(11), 1330–1334.