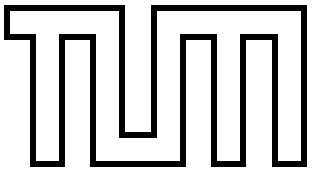


FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

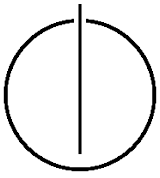
Master's Thesis in Computer Science

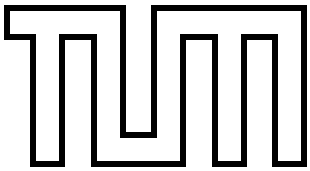
Out-of-Core Bundle Adjustment for 3D Workpiece Reconstruction

Robert Maier

In collaboration with Siemens Corporate Research Inc.

Princeton, New Jersey, USA





FAKULTÄT FÜR INFORMATIK

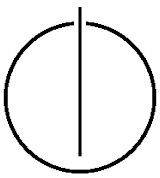
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Computer Science

Out-of-Core Bundle Adjustment for 3D Workpiece Reconstruction

Out-of-Core Bündelblockausgleichung für die 3D-Rekonstruktion von Werkstücken

Author:	Robert Maier
Supervisor:	Prof. Dr. Daniel Cremers
Advisor:	Dr. Jürgen Sturm
	Yao-Jen Chang, Ph.D.
Submission:	September 13, 2013



I assure the single handed composition of this master's thesis only supported by declared resources.

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

Munich, September 13, 2013 Robert Maier

Abstract

In this thesis, we developed an RGB-D-based 3D reconstruction framework with a special emphasis on out-of-core bundle adjustment. Our approach is suitable for 3D workpiece reconstruction as well as for the reconstruction of arbitrary scenes.

We first acquire RGB-D data of the scene by using a hand-held RGB-D sensor. The acquired depth map is preprocessed to achieve reduced sensor noise. Camera tracking is performed using a robust feature-based 3D alignment algorithm based on RANSAC. An initial map of the environment, which consists of camera poses, 3D landmarks and observations, is built using frame-to-frame tracking and is augmented with detected loop closures.

In order to reduce the effects of accumulated drift and inaccuracies in the map over time, bundle adjustment is applied. We minimize 3D alignment errors instead of 2D reprojection errors to improve robustness, convergence behaviour and accuracy. Since full bundle adjustment is unfeasible for large datasets, we introduce a novel combination of an efficient submap-based approach together with the minimization of 3D alignment errors. Our submap-based approach partitions the bundle adjustment problem into submaps, which are optimized independently of each other afterwards. After an efficient global alignment of the submaps, they are again optimized internally, but with fixed separators.

We finally integrate the RGB-D frames into an octree-based 3D representation to generate a dense 3D model.

Our submap-based bundle adjustment method significantly improves the runtime compared to full bundle adjustment, especially for large datasets. Further, our method approaches the accuracy of full bundle adjustment and outperforms the RGB-D SLAM system.

Finally, we present reconstructed 3D models of workpieces of compelling visual quality and metric accuracy, which makes them suitable for visual inspection, measuring tasks and reverse-engineering.

Acknowledgements

This thesis would not have been possible without the support of many people.

First of all, I want to thank Prof. Dr. Daniel Cremers for giving me the possibility to pursue my master's thesis at his chair.

This thesis would also not have been possible without the continuous support of my advisor Dr. Jürgen Sturm, who guided me throughout this thesis. His uncomplicated, friendly and open-minded attitude made working with him a real pleasure.

I am also very grateful to my supervisor at Siemens Corporate Reseach (SCR), Kevin Chang, who offered me the unique chance to do an exciting internship in Princeton, NJ. It was also him, who brought me to the challenging and fascinating topic investigated in this thesis.

The internship at SCR also gave me the chance to meet many great people from all around the world. Among them, I want to especially thank Ronny Bismark for proof-reading this thesis, for all the table tennis matches and for being a companion when we were both working late at SCR.

Further, I want to thank my family for all the support they gave me throughout my studies. Special thanks also go to my all of my friends, who were always there for me, and to my beloved Katharina for proof-reading this thesis and for all her support and patience over all the years.

Contents

1 Introduction

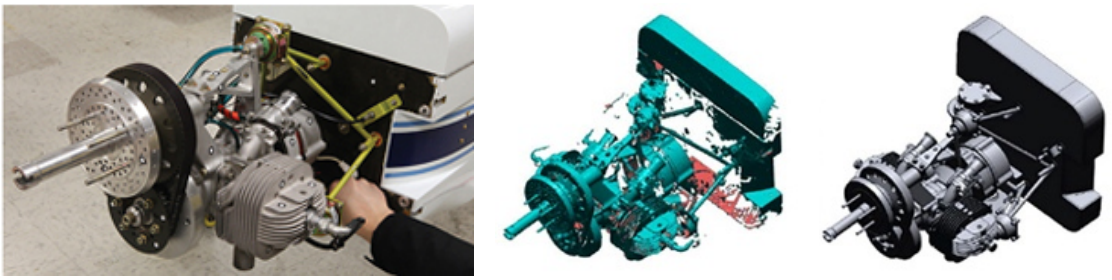
The camera-based reconstruction of digital 3D models from scenes and objects in the real world has been a challenging research topic in the field of Computer Vision over the last decades. The general approach is to build a complete 3D model of an object by fusing information from camera views from different perspectives into a global map representation. Here, the camera trajectory and the 3D model need to be estimated at the same time, which is also referred to as the problem of Simultaneous Localization And Mapping (SLAM). SLAM is the main concept to enable the 3D reconstruction of real world scenes. During the investigation of this research problem, a wide variety of application areas for 3D reconstruction has been found.

One of the most boosting application areas for 3D reconstruction can be found in the field of robotics. When a mobile robot platform wants to navigate in an unknown environment based only on information of a mounted camera, it needs to reconstruct a digital 3D model of the environment. This is important in order to guarantee self-localization and avoid collisions with objects in the environment in advance. Additionally, generating 3D models is also useful in other application scenarios such as gaming, physics or for medical purposes.

Besides the application scenarios mentioned above, reverse-engineering is one of the most important use cases of 3D reconstruction in the industrial field. In general, engineers often require 3D CAD models of real world objects for simulation or measuring purposes or even to reproduce real-world objects. These models should provide exact geometric information about the real objects, but the creation with common CAD software is complex and very time-consuming. Thus, automatic CAD model generation saves a lot of time and may even provide a higher accuracy than by manual creation. However, to generate CAD models from reconstructed 3D point clouds or meshes, further parametrization steps, usually provided by specialized software, are still needed. Figure 1.1 shows the results of a case study of reverse-engineering an engine [4].

The recent introduction of the affordable Microsoft Kinect RGB-D sensor has even increased the research interest in the field of 3D reconstruction, since it provides color images and depth values for each pixel at real-time frame rates. These accurate colored 3D point clouds of the scene, which are directly produced by the sensor, have been integrated into the approach of SLAM and so the process of creating metric 3D models of a scene has been expedited.

In order to create a complete dense 3D model of an object with high accuracy, which is particularly important for 3D reconstruction purposes, information from different camera views is required. This can be achieved by moving a hand-held camera around the object of interest and acquiring complementing scans from different perspectives. Camera tracking estimates the relative movement between the current and a previous camera pose and allows to fuse data from different views into a single global model representation. Both reconstruction and camera tracking combined, the reconstruction of geometrically precise



(a) (b) (c)

Figure 1.1: An example case study of reverse-engineering an engine (a) by using raw scan data (b) to create a CAD model (c). [4].

and complete 3D models is even possible in real-time, as recent approaches have shown.

1.1 Motivation

Among many possible applications of 3D reconstruction for reverse-engineering, we are especially interested in the 3D reconstruction of workpieces, which has many practical advantages.
First, generating an accurate colored 3D point cloud from a workpiece allows a thorough visual inspection of its surface without physical access of the inspecting person. The reconstructed model needs to be dense in order to show as many details of the surface as possible. In figure 1.2, a detailed 3D model of a lawn tractor, which has been reconstructed using our approach, is depicted.
A metrically accurate reconstruction allows measurements of the digital model. This can be easier than measurements of the real object.
By comparing the reconstruction with an already existing CAD model within an automatic inspection, deformations of the workpiece can be determined.
While there are no real-time requirements in the workpiece reconstruction scenario of this thesis, the process should nevertheless be as fast and efficient as possible.

Within the scope of this thesis, we assume that the scenes to be reconstructed are always static without moving or deforming objects. Furthermore, the scene must not be manipulated to facilitate the reconstruction. In practice, this means that there are no markers placed inside the scene and also no additional infrastructure (e.g. fixed mounted hardware) to support the reconstruction process.

To generate 3D models of workpieces, the 3D reconstruction system tracks the camera pose for each frame acquired from the RGB-D sensor, which allows to merge the frame data into a global 3D model. This way, a complete model can be built from many small frames. However, the estimation of the camera poses is subject to small errors based on uncertain noisy measurements. These small errors are accumulated due to tracking from frame to frame. This may finally result in inaccuracies and an accumulated drift over time in the estimated camera poses, which again causes a quality loss of the reconstructed 3D model.

Figure 1.2: The 3D model of a lawn tractor, which was reconstructed using the approach developed in this thesis. Such workpiece reconstructions allow visual inspections as well as metric measurements.

To account for these inaccuracies, the 3D model is optimized using bundle adjustment, where both estimated camera poses and structure (consisting of 3D feature points) are refined in order to reduce drift and to guarantee a metrically correct reconstruction of high accuracy.

The availability of RGB-D data allows to integrate additional depth constraints into the optimization procedure. However, the complexity of this non-linear optimization problem w.r.t. memory and computation time strongly depends on the number of frames and 3D points to be optimized. The solution of the bundle adjustment problem may consequently become problematic for the challenging amount of data needed to provide an accurate dense 3D model of a workpiece. In order to make bundle adjustment more efficient, an alternative out-of-core approach can be investigated. This submap-based bundle adjustment approach, which partitions the bundle adjustment problem into submaps, tries to improve efficiency, processing time and memory consumption, while at the same time maintaining global accuracy.

1.2 Problem statement

The objective of this thesis is to develop an RGB-D based 3D reconstruction framework that is able to reconstruct accurate dense 3D models of workpieces. A special focus is laid on achieving globally optimal reconstructions using bundle adjustment techniques. In particular, the contributions of this thesis are:

- A flexible and modular SLAM system based on RGB-D data is developed. While it is designed to suit the special case of 3D workpiece reconstruction, the reconstruction algorithm is held general to make it applicable to other general SLAM datasets and to reconstruct arbitrary scenes.
- To obtain globally accurate 3D reconstructions, out-of-core optimization techniques are applied. By integrating depth measurements as additional constraints into a submap-based bundle adjustment approach, the optimization goal is to minimize the 3D alignment error instead of the 2D reprojection error. The proposed novel approach results in a more efficient optimization and allows to perform bundle adjustment also on computers with limited hardware capabilities.
- A detailed evaluation of the developed RGB-D SLAM system and the implemented submap-based optimization approach is provided.

1.3 Outline

The outline of this thesis is structured as follows: First, **Chapter 2** exposes the current state of the art and gives an overview of existing related SLAM and bundle adjustment approaches.
Chapter 3 explains the mathematical tools and techniques required as preliminaries for the RGB-D SLAM system and the bundle adjustment approach developed in this work.
In **Chapter 4**, the components of the actual 3D reconstruction framework are described in detail. This includes also the out-of-core bundle adjustment approach, which allows to efficiently perform a global optimization for arbitrary RGB-D-based reconstruction problems.
In **Chapter 5**, an extensive quantitative evaluation of the defined 3D reconstruction system and of the implemented submap-based bundle adjustment approach w.r.t. processing time and accuracy is provided. Moreover, we show qualitative results of reconstructed workpieces.
Finally, **Chapter 6** gives a conclusion that summarizes the developed approach. Possible future extensions and improvements to this approach are pointed out in a final outlook.

2 Related Work

The 3D reconstruction of objects is derived from the well-investigated approach of SLAM with a special emphasis on building dense 3D models. Since SLAM problems have the main challenge to simultaneously estimate camera trajectory and a map of the environment based on noisy sensor data, inaccuracies in the computed camera poses and the map can occur. Therefore, bundle adjustment has been developed as a special optimization technique to solve this issue.

In the following, an overview of the state of the art of RGB-D SLAM and 3D reconstruction algorithms is given together with different bundle adjustment approaches.

2.1 SLAM and 3D reconstruction

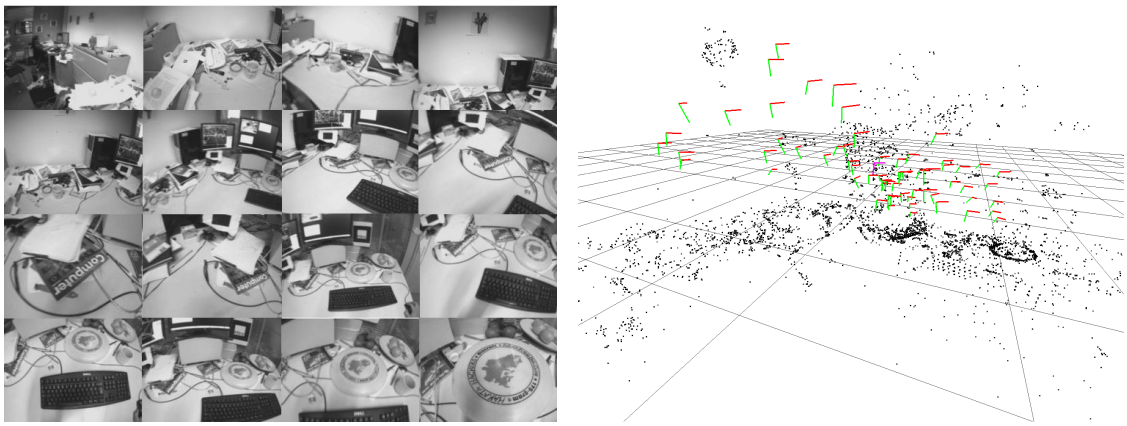
The general problem of SLAM has a quite long history in the field of robotics and has been well investigated since then. At the heart of the problem, the pose of a camera is continuously estimated w.r.t the environment. By utilizing data from a camera, a map of the environment is built, which at the same time is used to estimate the following camera poses.

Mostly, 3D mapping systems are based on a similar pipeline. After acquiring either color images, 3D information or a combination of both (RGB-D) as input data, sparse feature points are extracted from color images and used to perform a spatial relative 3D alignment between frame pairs. Alternatively, scan alignment using only 3D data can perform this alignment based on the Iterative Closest Point (ICP) algorithm. Afterwards, loop closures of the camera trajectory have to be detected. Finally, the data frames have to be aligned consistently w.r.t. a global coordinate system representing the model.

Over the years, many 3D mapping approaches using the data of only a single RGB camera have been developed. This monocular SLAM scenario, which is also referred to as Structure from Motion (SfM), is covered in approaches like [14], MonoSLAM [15] or PTAM [25]. They perform efficient camera tracking and mapping in real-time, but tend to produce sparse maps of the environment as shown in figure 2.1.

These maps can only be determined up to scale and thus don't allow a reconstruction in real metric units. Furthermore, while these methods are quite accurate for camera tracking, they are only of limited use for obtaining dense 3D models. To overcome this, more recent approaches use every pixel of an input image for camera tracking (instead of sparse feature points) and try to generate dense maps [32, 34].

As recently low-cost RGB-D sensors, in particular the Microsoft Kinect, have been introduced, these sensors have been utilized to resolve some of the challenges of monocular SLAM (e.g. metric measurements). RGB-D cameras capture RGB images with per-pixel depth information at real-time frame rates. The developed RGB-D SLAM approaches combine visual information from the color images with 3D information from the depth mea-



(a) (b)

Figure 2.1: Monocular SLAM approaches such as PTAM usually produce sparse maps, which can be determined only up to scale. (a) A subset of input frames of a desk and (b) the generated map with point features and camera poses. [25]

surements to create dense 3D environment representations. Since RGB-D sensors provide the depth in real metric units, it is possible to directly build metrically correct 3D reconstructions of objects, which is also the reason that we focus on RGB-D based approaches in the following.

2.1.1 KinectFusion

The KinectFusion system developed by Newcombe et al. [33, 24] uses a single hand-held Kinect to reconstruct arbitrary complex, room sized indoor scenes in real-time. The high performance of the system is achieved by an extensive use of highly parallel general purpose GPU programming techniques.

At the heart of the approach, the reconstructed model is stored in a global volumetric representation in the form of a 3D voxel cube. Here, all depth measurements acquired by the sensor are fused into an implicit surface model, known as Truncated Signed Distance Function (TSDF) volume, of the observed scene in real-time. Figure 2.2 shows a 3D model reconstructed using KinectFusion.

Camera tracking is performed by estimating the current camera pose w.r.t. to the full dense TSDF model at 30 Hz frame-rate. Therefore, the current camera frame is aligned to a frame of the model, which is generated synthetically by ray-casting the TSDF, using a fast ICP algorithm in coarse-to-fine manner. Using this estimated camera pose, the current frame is integrated into the TSDF, so that the 3D surface model is continuously up-to-date, smooth, and fully fused. An overview of the full system pipeline of the KinectFusion system is presented in figure 2.3.

The reconstructed dense 3D model shows limited drift and is characterized by a beforehand unseen high accuracy due to the use of the TSDF representation. However, its quality depends on the size of the voxels in the volumetric cube and is in practice limited by the amount of memory of the graphics card. This fixed-size voxel model is also considered

Figure 2.2: RGB image (A), surface normals (B) and noisy surface reconstruction (C) from a single bilateral filtered input frame. (D) and (E) show the surface normals and the Phong shaded model of a 3D model generated from KinectFusion. [24].

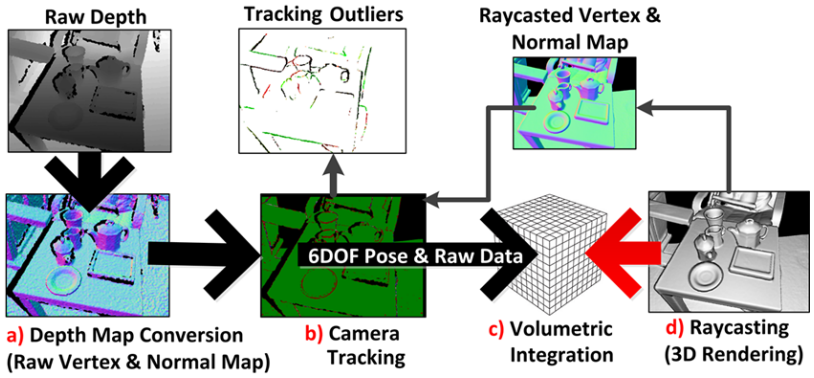


Figure 2.3: The main stages of the KinectFusion tracking and reconstruction pipeline [24].

as the main drawback of KinectFusion, since only small scenes can be represented in the model volume at a sufficiently high resolution. Furthermore, the system cannot recover from failed ICP-tracking caused by poor 3D geometry in the scene.

2.1.2 RGB-D Mapping

While KinectFusion is rather targeted at the real-time reconstruction of room-sized models, the approach of RGB-D Mapping of Henry et al. [23] concentrates on the mapping of large indoor environments. The overall structure of this approach (figure 2.4) is designed similarly to established SLAM techniques, augmented by the use of RGB-D sensors.

For camera tracking, a combined approach called RGBD-ICP was developed to provide a relative alignment of consecutive frames. This approach first performs a relative alignment based on sparse feature point correspondences of the color images. This first feature-based alignment is then used as an initialization to an ICP-based pose estimation that relies on depth information. The detection of loop closures is implemented by matching RGB features between the current frame and some previous frames, which are spatially close. In the case of a successful matching, the relative pose between the respective frames is estimated and added to the pose graph. For global consistency of the created map, a final pose graph optimization is performed.

Using the computed global alignment, a frame is integrated into a global Surfel-based

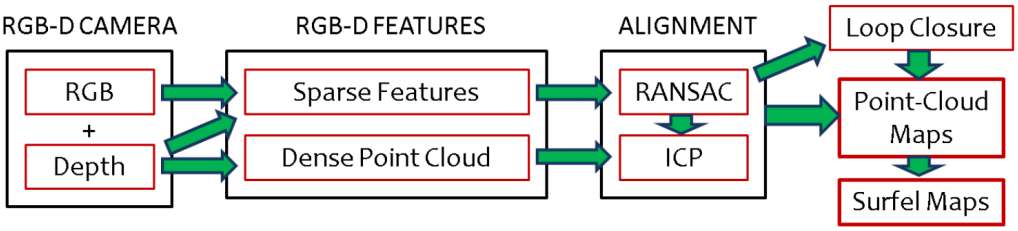


Figure 2.4: The RGB-D Mapping algorithm combines sparse feature-based and dense ICP-based 3D alignment for relative pose estimation. [23].

3D model representation. This dense representation can be stored efficiently and provides a smooth visualization.

All in all, RGB-D Mapping has the disadvantage of limited use for real-time applications, because it does not exploit GPU hardware. Moreover, more consistent reconstructions could be achieved by including also 3D landmarks into the optimization.

2.1.3 RGB-D SLAM

With the RGB-D SLAM system by Endres et al. [16, 17], a system very similar to RGB-D Mapping has been introduced. However, its basic approach depicted in figure differs in camera tracking and the chosen model representation.

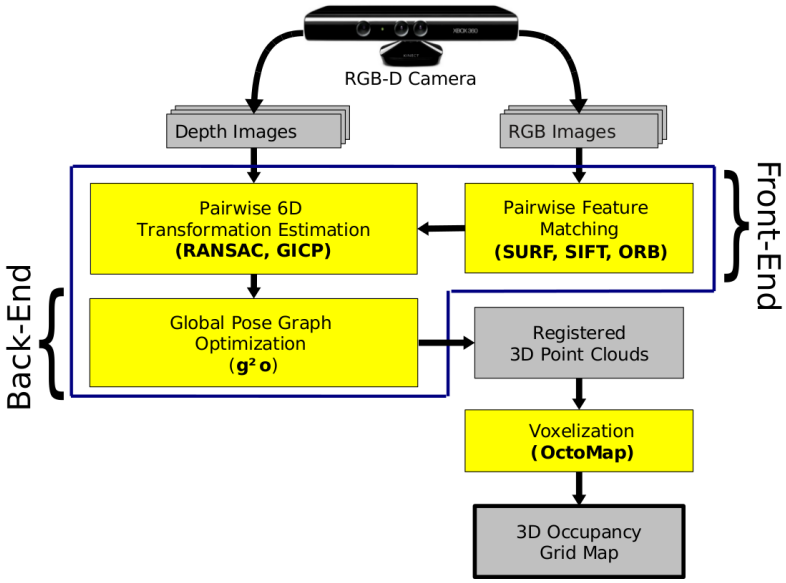


Figure 2.5: Schematic overview of the RGB-D SLAM system. The SLAM front-end estimates relative poses between the frames using feature-based 3D alignment, while the SLAM back-end performs a pose graph optimization [16].

The localization of the camera is computed similarly to RGB-D mapping, but in a simplified manner. For each new input RGB image acquired from the RGB-D sensor, visual fea-

tures are extracted and matched with the features of the color image of the previous frame. Using the depth information of the feature matches, point-wise 3D correspondences between the frames can be established. These 3D correspondences allow to robustly estimate the relative 3D alignment between the frames using RANSAC. Loop closures are detected in the same way as it is done in RGB-D Mapping. Since the realized frame-to-frame tracking results in an accumulated drift in the estimated camera poses, global consistency is achieved by optimizing the pose graph using the *g2o* framework for optimizing graph-based nonlinear error functions [27].

As output of the system, a dense colored 3D point cloud is generated from an octree-based model representation. This efficient volumetric representation is realized using the Octomap library [50], which explicitly represents free space and unmapped areas. The system is extensively evaluated using the recently published TUM RGB-D benchmark datasets by Sturm et al. [44].

While the RGB-D SLAM system is characterized by its highly flexible processing pipeline based on robust feature-based camera tracking, full bundle adjustment could improve the global accuracy.

2.2 Bundle adjustment

The related work shown in the previous section exhibited a special focus on RGB-D-based SLAM systems, which allow dense 3D reconstruction of scenes. Except of KinectFusion, which uses the TSDF model representation of limited size, all the presented systems create and maintain an internal map of the environment similar to a graph representation. Uncertainties and inaccuracies in the relative pose estimation deteriorate the quality and global consistency of the map and require a subsequent optimization. For this reason, SLAM approaches usually contain a bundle adjustment step, which takes care of this global optimization and is important for the quality of the reconstructed 3D models. From the different kinds of bundle adjustment methods, we consider the following ones as most important:

- Full bundle adjustment
- Pose-graph optimization
- Active window approaches
- Submap-based approaches

While online bundle adjustment approaches emphasize locally accurate reconstructions, offline (or batch) methods account for global consistency and are mostly not computable in real-time.

In the following, the types of bundle adjustment mentioned above are treated by showing the most relevant related work of each type. Hereby, the capability of optimizing large datasets, as it can occur in the reconstruction of workpieces, serves as a special motivation.

2.2.1 Full bundle adjustment

Bundle adjustment was introduced into the Computer Vision community by Triggs et al. [47] in order to optimize the results obtained in SLAM systems. It is described in detail

in section 3.4. The problem to be optimized consists of camera poses with six degrees of freedom, 3D landmarks and observations that usually describe the 2D position of the landmarks in the respective camera poses. When we represent these entities in a graph, camera vertices are connected only to landmark vertices by the observations as edges. This structure is usually the internal map in SLAM systems and also called SLAM graph. Traditional bundle adjustment performs a graph optimization of the SLAM graph by minimizing the 2D reprojection errors of the landmarks in the camera frames based on the 2D observations. Because this is a Non-linear Least Squares (NLS) problem, it can be solved using the Levenberg-Marquardt (LM) algorithm; the parameters to be optimized are the camera poses and the 3D landmark locations. As a result, a metrically correct global reconstruction is obtained.

In full bundle adjustment, the optimization is usually performed as a batch process after all the available camera frames have been aligned and the complete map representation is constructed. The full map, consisting of all camera poses, all 3D landmarks and all the connecting observations, is used as input problem. However, both time and space complexity of such traditional bundle adjustment algorithms are cubic in the number of variables, which is especially problematic in large datasets. In SLAM problems, the sparse connections between camera poses and landmarks can be exploited. This method of sparse bundle adjustment has a still high computational complexity, which is at least cubic in the number of camera poses plus a linear complexity in the number of landmarks. Triggs et al. [47] give a very detailed survey on sparse bundle adjustment with descriptions of its mathematical background, computational considerations and NLS optimization methods.

2.2.2 Pose graph optimization

While the optimization problem in full bundle adjustment consists of camera poses, landmarks and observations, some works reduce the graph to contain only poses and pose-pose-connections [16, 23, 26]. Such a graph representation is usually called pose graph, which consists only of the camera trajectory. The landmarks are marginalized out, since they are already used for estimating the relative poses between frames. Thus, the vertices of the optimization graph are only the camera poses, while the estimated relative poses are the edges that connect them. The edges directly represent the motion between consecutive frames or detected loop closures.

Because of the reduced size of the optimization problem, pose-only optimization is generally more efficient than full bundle adjustment. Simplified, pose-only optimization concentrates more on the camera trajectory than on a complex map. Moreover, maintaining and updating the map is simpler due to the absence of landmarks and their observations. For detected loop closures, when previously visited areas are recognized again, it is substantially more efficient to close large loops in pose graphs than in full bundle adjustment.

However, it can be seen as a drawback that the non-linear connections between poses and points are not fully encoded in the binary links between the camera poses. The pose-pose reduction is hence an approximation per se and the subsequent pose graph optimization does not achieve the same accuracy as full bundle adjustment.

2.2.3 Active window approaches

Due to the considerable amount of data involved in the optimization of the full SLAM graph at once, there has been research in performing bundle adjustment only on a smaller portion of the problem. The goal is to still get a globally optimal reconstruction while allowing to perform bundle adjustment online during the mapping process. In active window approaches, the optimization is usually computed only based on a small sub-graph. This so-called window consists of the neighborhood around the current frame and "slides" with every new frame added.

Engels et al. [18] have presented an incremental bundle adjustment process, which performs bundle adjustment every time after a new frame has been added to the SLAM graph and still guarantees online operability in constant-time. Here, the graph to be optimized consists of a sliding window of the most recently added frames and the observed landmarks, while the frames added beforehand are fixed. The global drift can be limited significantly, but since no loop closures are detected, global consistency cannot be guaranteed.

To head towards finding a globally correct solution, Strasdat et al. [42] developed a double window optimization framework. Their algorithm dynamically selects a subset of all keyframes as an active window, which is again dynamically split up into an inner and outer window based on the covisibility of shared common features between frames. The inner window consists of the region around the current frame to be added and is modeled by pose-point constraints between the contained camera frames and landmarks, as in regular bundle adjustment. This inner window is supported by an outer window modeled by a set of pose-pose constraints like in pose graph optimization. A sample SLAM graph and its determined active window is depicted in figure 2.6.

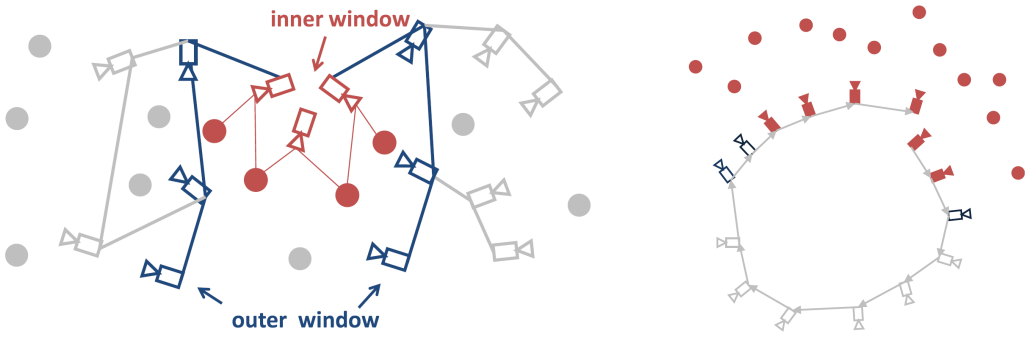


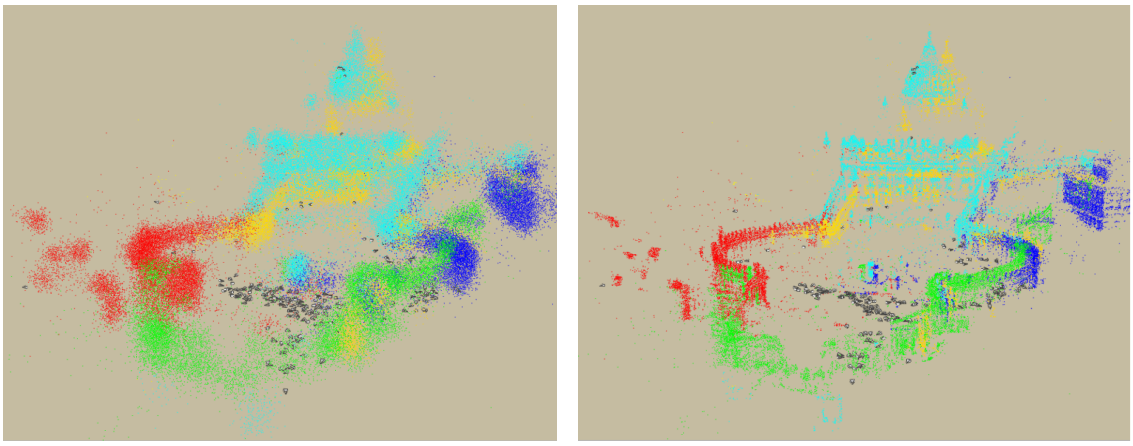
Figure 2.6: Example SLAM graphs for the double window approach. In (a), an active window around the current frame is divided into an inner (red) and an outer (blue) window and inactive frames (gray). (b) shows a loop closure in the case of a loopy camera motion. [42]

While the inner window is used to provide an accurate metric representation in the local area, the outer window serves to stabilize the periphery. The presented active window scheme has the advantage, that it achieves constant-time performance in the case of a fixed size of both windows. It takes a locally accurate reconstruction into account as well as loop closures in a larger scale. However, the results have shown that an overall global metric

mapping, comparable to full offline bundle adjustment, can only be obtained by including all frames, which are not in the inner window, into the outer window. This extension of the outer window comes along with a cubic complexity in the number of frames and is again computationally expensive.

2.2.4 Submap-based approaches

Since in full bundle adjustment the factored sparse matrices of large problems may exceed the core memory, it is reasonable to perform bundle adjustment on several smaller parts of the entire scene. While this is also the goal of active window approaches, these methods however do not guarantee a global metric reconstruction. Nevertheless, the full problem can also be partitioned into several smaller submaps, which are connected with each other. These divide-and-conquer methods are known as submap-based approaches (or submapping) and can be solved quite efficiently. An example of such an approach can be seen in figure 2.7. Improved efficiency is accomplished by optimizing the submaps consecutively in a parallel or even in an out-of-core manner, when only the current submap of interest has to be loaded into memory. The other submaps could be stored in files on the hard disk. Adjusting the number and size of the submaps allows to increase the scalability w.r.t. overall computation time and memory when the entire scene has to be bundle adjusted. Traditional bundle adjustment can be seen as a special case of submapping, when all the parameters are combined into a single submap.



(a) Unoptimized map. (b) Map after optimization.

Figure 2.7: In submap-based approaches the entire map is partitioned into several submaps (represented by different colors), which are first optimized independently and then combined by a global alignment. [37]

Ni and Dellaert presented some work on submap-based 3D reconstruction [37, 38], which is based on a very efficient out-of-core bundle adjustment algorithm. In general, submap-based approaches can be decomposed into the following steps:

1. Partitioning of the full problem into connected submaps with as few dependencies between the submaps as possible.

2. Full bundle adjustment of the individual submaps.
3. Global optimization by joining all the optimized submaps.

The concrete form of the steps above can vary depending on the respective submapping approach. In the following, the almost identical approaches of Ni and Dellaert [37, 38] are explained in more detail.

Submap partitioning: First, the full SLAM graph is partitioned into a number of submaps. The submap size affects both optimization time and the global reconstruction accuracy. Smaller submaps result in an expensive global, while larger submaps are individually more expensive to optimize.

Measurements depending on parameters in different submaps are called inter-measurements, all other measurements are intra-measurements. The submap graph nodes contributing to inter-measurements are boundary/separator variables and all other variables are internal variables. Figure 2.8 shows an example SLAM graph, which is partitioned into two submaps.

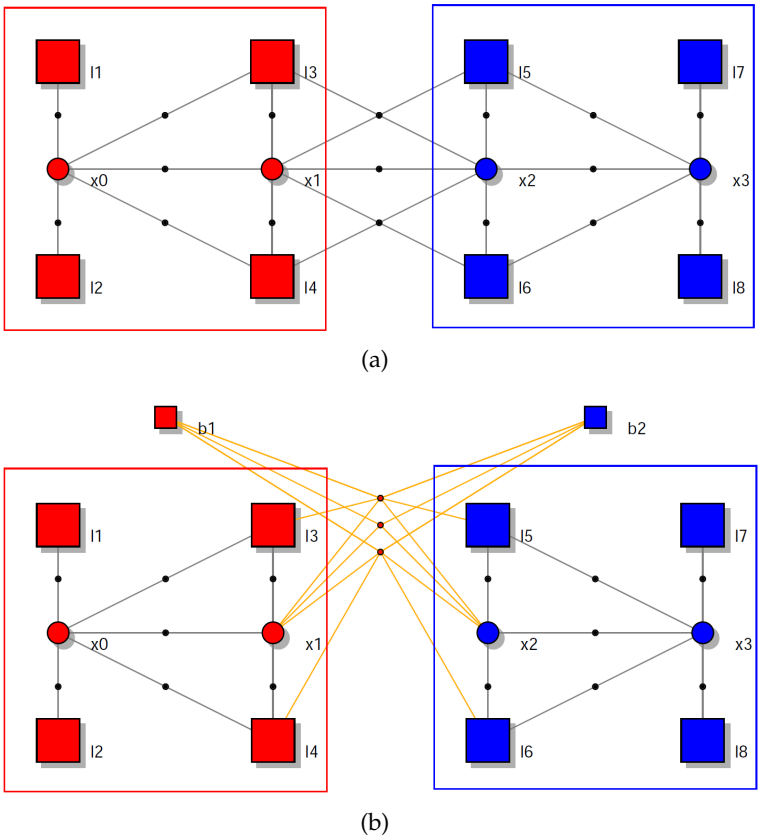


Figure 2.8: (a) A SLAM graph with four camera poses (circles) and eight landmarks (squares) partitioned into two (differently colored) submaps. (b) Base nodes are added to each submap, which are connected by inter-measurements (yellow). [38]

Optimally, the graph partition itself should minimize the number of measurements between the different submaps and lead to a small number of separator variables. Here, the graph cut is determined by nested dissection. In this divide-and-conquer approach the parameter network in a sparse system of equations is directly partitioned into several submaps.

To optimize the submaps independently, a base node is assigned to each submap and the submap's poses and landmarks are expressed relative to this base node. This concept of relative bundle adjustment has been introduced by Sibley et al. [40]. Base poses and boundary variables from all submaps combined represent the separating set of the overall problem.

Optimization: With the original problem partitioned into submaps with their own local coordinate systems, the algorithm in [37] performs the optimization by iterating over three stages:

1. The internal variables in each submap are factored out and their linearizations are cached. The resulting reduced system of equations depends only on the separator variables. In this stage, the processing of the individual submaps can be done in parallel.
2. The introduction of base nodes for each submap allows the non-linear optimization to converge significantly faster. Under the assumption that the submaps have been optimized adequately, they only have to be moved globally by undergoing rigid transformations with respect to each other. Therefore, the separator variables are connected to the based nodes and optimized using the cached linearizations of the intra-measurements. At each iteration of this global alignment, the inter-measurements are relinearized again. The global optimization step itself moves only the base nodes instead of the actual submap variables, which automatically moves all the variables contained in a submap. The relative parametrization guarantees that the cached linearizations of the intra-measurements stay accurate even when the base nodes are moved significantly.

   This global alignment method makes it possible to globally join submaps without the requirement, that the full bundle adjustment problem has to be in core memory at once. After the global submap alignment, the separator variables are optimal up to the intra-measurements linearization.
3. In the third stage, the internal variables in the submaps are optimized by performing an optimization of each submap with locked separator variables.

The presented algorithm is able to convergence to an optimal solution in only two iterations. This is of importance in the out-of-core variant of the algorithm, since each submap has to be loaded from hard disk into memory in each iteration.

Instead of iterating over the three stages above, the *Tectonic SAM* approach of Ni and Dellaert [38] performs the stages only once without iterating. However, the stages themselves are designed differently:

1. Each submap is optimized locally and independently by integrating intra-measurements only. This approach makes only few assumptions about the required quality

of the graph partitioning.

2. Again, since base nodes for each submap are introduced for relative parametrization, the separator is optimized using only inter-measurements. Therefore, the boundary nodes in the separator keep their values from the internal submap optimization, while the base nodes have to be initialized based on the camera odometry. At each iteration of the separator optimization, only the inter-measurements are relinearized. The relative parametrization again has the effect, that large global transformations of the submaps significantly influence the linearization point of the inter-measurements and only marginally affect the intra-measurements. In practice, the number of separator variables is larger than any of the submap graphs and therefore more time-consuming.
3. To back-propagate the changes of the separating set after the separator optimization has converged, the nodes in each submap are finally updated by back-substitution. After the final values of the non-boundary nodes in the submap are computed, the final solution of the submap-based approach is obtained.

The two explained submapping approaches [37, 38] are suitable for online mapping scenarios, as already optimized submaps can provide local reconstructions of high quality. However, scenarios with many loop closures (such as unstructured photo collections) can be less efficient and hence problematic for online mapping.

Recently, Ni and Dellaert extended their work by recursively partitioning the SLAM graph into multiple-level submaps [35, 36] and by applying adequate bottom-up optimization approaches. They investigated SLAM graph partitioning based on nested dissection to recursively obtain small submap separators.

Because pose graph and active window approaches do not consider all information available, they do not guarantee a global metric reconstruction. Submap-based bundle adjustment methods however approach the global accuracy of full bundle adjustment paired with a substantially more efficient computation. This makes submap-based methods especially suitable for the bundle adjustment approach of this thesis, as they can significantly improve the efficiency of bundle adjustment in the workpiece reconstruction scenario.

3 Preliminaries

This chapter provides the mathematical background that will be used in the subsequent parts of this thesis and is based on [22, 46]. However, as some of the provided concepts cannot be covered exhaustively, we refer the reader to the referenced literature for deeper insights. The reader is further assumed to have a basic knowledge of linear algebra. Bundle adjustment is presented in a manner similar to [28].

3.1 Points and transformations

Points as the most important geometric primitives are denoted by bold letters and are represented by vectors. A point $\mathbf{x}$ in 2-space is represented by a vector $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$ or in homogenous coordinates by a 3-vector $\bar{\mathbf{x}} = (x, y, 1)^\top$. Similarly, a 3D point $\mathbf{X}$ can be denoted by a vector $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ or in homogenous coordinates by a 4-vector $\bar{\mathbf{X}} = (X, Y, Z, 1)^\top$.

A **3D Euclidean transformation** consists of rotation and translation in 3D space and thus has six degrees of freedom. Because it preserves distance and orientation of any point pair of a rigid body, it is also known as "rigid body motion". The set of rigid body motions forms a Lie group, namely the Special Euclidean group $SE(3)$, and is described in detail by Ma et al. [30]. While the translational part of a rigid body motion $g = (R, t) \in SE(3)$ is defined by a translation vector $t \in \mathbb{R}^3$, there are different representations for the rotational part $R \in SO(3)$, where $SO(3)$ is the special orthogonal group representing rotations. In this thesis we will only use 3×3 rotation matrices and twist coordinates to represent rotations.

An orthogonal 3×3 rotation matrix $R \in \mathbb{R}^{3\times3}$ (with $RR^\top = I$ and $|R| = 1$) contains nine parameters and is an overparametrized representation of a 3D rotation consisting of only three degrees of freedom. Both rotation matrix R and translation vector t can be combined into a 4×4 transformation matrix to describe a rigid body motion $g \in SE(3)$:

$$T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

Further, we can define a transformation $\mathcal{T}$ of a 3D point $\mathbf{P} = (X, Y, Z)^\top \in \mathbb{R}^3$ to a transformed 3D point $\mathbf{P'} = (X', Y', Z')^\top$:

$$\mathcal{T} : SE(3) \times \mathbb{R}^3 \to \mathbb{R}^3, \mathbf{P'} = \mathcal{T}(g, \mathbf{P}) = R\mathbf{P} + t = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_X \\ t_Y \\ t_Z \end{pmatrix}. \tag{3.2}$$

The inverse transformation $\mathcal{T}^{-1}$ that maps $\mathbf{P}'$ back to $\mathbf{P}$ is given by:

$$\mathcal{T}^{-1} : SE(3) \times \mathbb{R}^3 \to \mathbb{R}^3, \mathbf{P} = \mathcal{T}^{-1}(g, \mathbf{P}') = R^\top(\mathbf{P}' - t). \tag{3.3}$$

A homogenous 3D point $\bar{\mathbf{P}} = (X, Y, Z, 1)^\top \in \mathbb{R}^4$ can be transformed to $\bar{\mathbf{P}}' = (X', Y', Z', 1)^\top$ using multiplication with a 4×4 transformation matrix:

$$\mathcal{T}' : SE(3) \times \mathbb{R}^4 \to \mathbb{R}^4, \bar{\mathbf{P}}' = \mathcal{T}'(g, \bar{\mathbf{P}}) = T\bar{\mathbf{P}}. \tag{3.4}$$

By exploiting the identity of $R^{-1} = R^\top$, the inverse T^{-1} of a 4×4 transformation matrix can also be written as:

$$T^{-1} = \begin{bmatrix} R^\top & -R^\top \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.5}$$

which can be used to construct the inverse transformation of homogenous 3D points accordingly:

$$\mathcal{T}'^{-1} : SE(3) \times \mathbb{R}^4 \to \mathbb{R}^4, \bar{\mathbf{P}} = \mathcal{T}^{-1}(g, \bar{\mathbf{P}}') = T^{-1}\bar{\mathbf{P}}'. \tag{3.6}$$

The advantage of using homogenous matrices in the form $T \in \mathbb{R}^{4\times4}$ is that consecutive transformations can be easily combined into a single transformation matrix by multiplying the respective transformation matrices. This allows us to transform a transformation $g_2 \in SE(3)$ using another transformation $g_1 \in SE(3)$ by multiplication of their transformation matrices $T_1, T_2 \in \mathbb{R}^{4\times4}$ to get the combined transformation $g_3 \in SE(3)$:

$$\mathfrak{T} : SE(3) \times SE(3) \to SE(3), g_3 = \mathfrak{T}(g_1, g_2) = T_1 T_2. \tag{3.7}$$

The inverse transformation is defined as:

$$\mathfrak{T}^{-1} : SE(3) \times SE(3) \to SE(3), g_3 = \mathfrak{T}^{-1}(g_1, g_2) = T_1^{-1} T_2. \tag{3.8}$$

Instead of using rotation matrices, a 3D Euclidean transformation $g \in SE(3)$ can also be represented by 6-dimensional twist coordinates:

$$\xi = (\omega_1, \omega_2, \omega_3, v_1, v_2, v_3)^\top \in \mathbb{R}^6 \tag{3.9}$$

As only six parameters are used to represent a 6-DOF motion, this is a minimal representation and suitable for numerical optimization. Here, rotation and translation are described by $\omega_\xi = (\omega_1, \omega_2, \omega_3)^\top \in \mathbb{R}^3$ and $\mathbf{t}_\xi = (v_1, v_2, v_3)^\top \in \mathbb{R}^3$ respectively. A conversion from twist coordinates to rotation matrices (and vice versa) is possible.

3.2 Camera model

The mapping of 3D points of a three-dimensional scene onto a 2D image plane is described by the camera model, which is represented by a matrix with particular properties. In this work we use the basic pinhole camera model depicted in figure 3.1. This model reflects the central projection of points onto a plane, where the camera center $\mathbf{C}$ is the center of projection. The camera coordinate system is an Euclidean coordinate system with $\mathbf{C}$ as its origin. The image plane is defined by $Z = f$, where f is the focal length. The principal

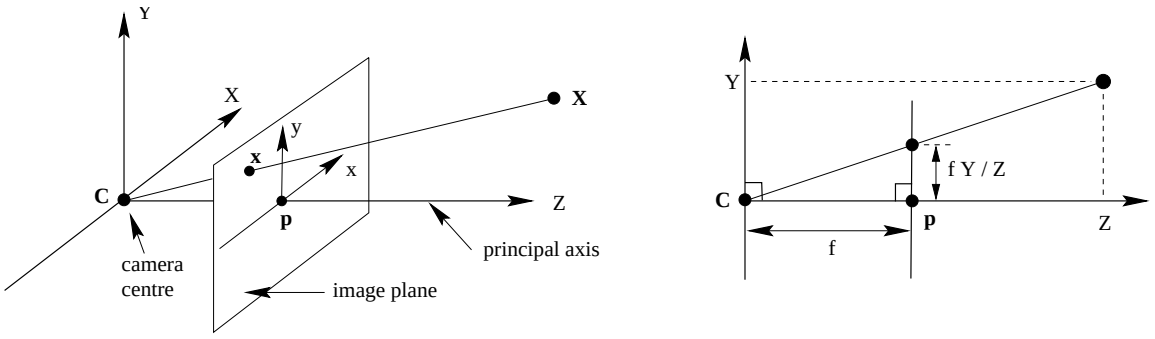


Figure 3.1: Pinhole camera model with camera center C, principal point p and focal length f [22].

axis crosses **C** and is perpendicular to the image plane; the point where the principal axis intersects with the image plane is called the principal point $\mathbf{p} = (p_x, p_y)^\top \in \mathbb{R}^2$.

The 2D image point $\mathbf{x} = (u, v)^\top \in \mathbb{R}^2$ of a 3D point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ is determined by the intersection of the line between **C** and **X** with the image plane. By similar triangles, the 2D image point can be computed as $\mathbf{x} = (\frac{fX}{Z}, \frac{fY}{Z})^\top$. Since the principal point is usually not the origin of the image coordinate system, the principal point offset needs to be considered:

$$\mathbf{x} = \left(\frac{fX}{Z} + p_x, \frac{fY}{Z} + p_y\right)^\top. \tag{3.10}$$

In the case of CCD cameras, the image coordinates are not measured in Euclidean coordinates, but in pixels. Moreover, the pixels are not necessarily squared, which all in all requires to introduce scale factors for each direction of the image coordinate frame. With m_x and m_y reflecting the number of pixels per unit distance in each direction, the focal length can be expressed as $f_x = fm_x$ and $f_y = fm_y$ in terms of pixel dimensions in x and y direction. The principal point coordinates in terms of pixel dimensions are $c_x = m_x p_x$ and $c_y = m_y p_y$.

The camera calibration matrix $K \in \mathbb{R}^{3\times3}$ contains the camera intrinsics and can finally be written as:

$$K = \begin{bmatrix} f_x & & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix}. \tag{3.11}$$

Multiplying the camera matrix K with an inhomogenous 3D point $\mathbf{X} = (x, y, z)^\top \in \mathbb{R}^3$ results in the corresponding projected 2D point $\bar{\mathbf{x}} \in \mathbb{R}^3$ in homogenous coordinates.

To obtain the projected 2D point directly in inhomogenous coordinates, we define the projection function π. Using the camera intrinsics, a 3D point $\mathbf{X} \in \mathbb{R}^3$ is mapped to a 2D image point $\mathbf{x} \in \mathbb{R}^2$ on the image plane as follows:

$$\pi : \mathbb{R}\times\mathbb{R}\times\mathbb{R} \to \mathbb{R}^2, \mathbf{x} = (u, v)^\top = \pi(x, y, z) = \left(\frac{f_x x}{z} + c_x, \frac{f_y y}{z} + c_y\right)^\top. \tag{3.12}$$

If a depth value d for a 2D image point $\mathbf{x} = (u, v)^\top \in \mathbb{R}^2$ is given, the back-projection of this point to a 3D point $\mathbf{X} \in \mathbb{R}^3$ in the camera coordinate frame can be computed as follows:

$$\rho : \mathbb{R}\times\mathbb{R}\times\mathbb{R} \to \mathbb{R}^3, \mathbf{X} = (x, y, z)^\top = \rho(u, v, d) = \left(\frac{(u-c_x)d}{f_x}, \frac{(v-c_y)d}{f_y}, d\right)^\top. \tag{3.13}$$

3.3 3D alignment estimation from 3D point correspondences

The problem of determining rigid body motions for camera tracking is one of the most critical parts in many SLAM algorithms. In feature-based 3D alignment methods, we usually determine correspondences of 3D points in different camera frames, which are used to estimate the relative pose between the two camera poses (see figure 3.2). In particular,

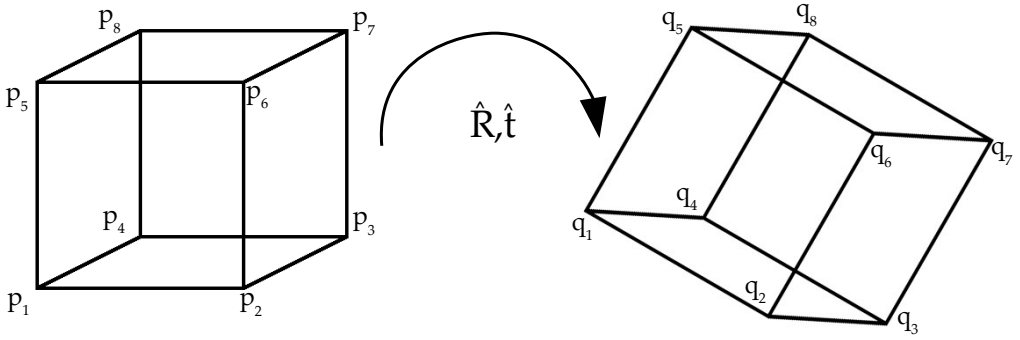


Figure 3.2: Given two sets of corresponding 3D points, we can compute the relative pose $(\hat{R}, \hat{\mathbf{t}})$ that aligns them.

the accuracy of the entire 3D reconstruction framework is determined by the estimation of these relative camera poses. A well-known non-iterative algorithm for estimating the 3D Euclidean transformation from 3D point correspondences is stated in [6] and explained in the following.

Given are two point sets $P = \{\mathbf{p}_i\}$ and $Q = \{\mathbf{q}_i\}$ with $i \in 1 \dots n$ and $n \geq 4$ points each. The points $\mathbf{p}_i \in \mathbb{R}^3$ and $\mathbf{q}_i \in \mathbb{R}^3$ are corresponding 3D points and are perturbed by noise $\mathbf{N}_i$ stored in a vector N. In this section it is assumed that all correspondences between the point sets are correct; if outliers are present a RANSAC-based approach is necessary.

These two point sets are related by a transformation $\hat{g} \in SE(3)$, which is represent using a rotation matrix $R \in \mathbb{R}^{3\times3}$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$:

$$\mathbf{q}_i = R\mathbf{p}_i + \mathbf{t} + \mathbf{N}_i. \tag{3.14}$$

To determine the optimal rotation $\hat{R}$ and the optimal translation $\hat{\mathbf{t}}$, we have to minimize the following least-squares problem:

$$\Sigma^2 = \sum_{i=1}^{n} \left\| \mathbf{q}_i - (\hat{R}\mathbf{p}_i + \hat{\mathbf{t}}) \right\|^2. \tag{3.15}$$

The algorithm to find the optimal rotation and translation is explained in the following:

Zero-centered point sets: In order to calculate the rotation, first the centroids (centers of mass) $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$ of both point sets are determined:

$$\bar{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i \quad \text{and} \quad \bar{\mathbf{q}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{q}_i. \tag{3.16}$$

Both point sets are translated to the origin of their respective coordinate frames by subtracting their centroid from every point. As results, we obtain the zero-centered points $\mathbf{p}_{c_i}$ and $\mathbf{q}_{c_i}$:

$$\mathbf{p}_{c_i} = \mathbf{p}_i - \bar{\mathbf{p}} \quad \text{and} \quad \mathbf{q}_{c_i} = \mathbf{q}_i - \bar{\mathbf{q}}. \tag{3.17}$$

Optimal rotation $\hat{\mathbf{R}}$: To find the rotation $\hat{R}$ that best aligns both zero-centered point sets, we rewrite the least-squares minimization problem:

$$\Sigma^2 = \sum_{i=1}^{n} \left\| \mathbf{q}_{c_i} - \hat{R}\mathbf{p}_{c_i} \right\|^2 = \sum_{i=1}^{n} \left\| \mathbf{q}_{c_i}^\top \mathbf{q}_{c_i} + \mathbf{p}_{c_i}^\top \mathbf{p}_{c_i} - 2\mathbf{q}_{c_i}^\top \hat{R}\mathbf{p}_{c_i} \right\|^2. \tag{3.18}$$

Minimizing equation 3.18 can be achieved by maximizing the term containing $\hat{R}$. By defining the matrix $H \in \mathbb{R}^{3\times3}$ as

$$H = \sum_{i=1}^{n} \mathbf{p}_{c_i} \mathbf{q}_{c_i}^\top, \tag{3.19}$$

this is equivalent to maximizing the trace $Tr(\hat{R}H)$. Using the Singular Value Decomposition (SVD) $H = U\Sigma V^\top$, the trace is maximized by setting the rotation matrix to

$$\hat{R} = VU^\top. \tag{3.20}$$

If the determinant $\det(\hat{R})$ is $+1$, then the solution is unique and we have found the optimal rotation. If $\det(\hat{R})$ is -1, then $\hat{R}$ is a reflection, which can occur when much noise is present or the point sets are co-planar. In this case, the rotation is computed as $\hat{R} = V'U^\top$, where V' is obtained from V by changing the sign of the third column vector.

Optimal translation $\hat{\mathbf{t}}$: Once $\hat{R}$ is determined, the optimal translation $\hat{\mathbf{t}}$ is finally computed as the difference between the centroid $\bar{\mathbf{p}}$ and the rotated centroid $\bar{\mathbf{q}}$:

$$\hat{\mathbf{t}} = \bar{\mathbf{p}} - \hat{R}\bar{\mathbf{q}}. \tag{3.21}$$

As we have determined the optimal rotation and translation, we have also obtained the optimal relative pose $\hat{g} = (\hat{R}, \hat{\mathbf{t}}) \in SE(3)$ between the two 3D point sets.

3.4 Bundle adjustment

In the case of 3D reconstruction based on SfM, 2D image measurements from multiple images are utilized to reconstruct the 3D structure of a scene as well as the camera trajectory and the camera intrinsics. In particular, a map of the scene is stored internally with n 3D landmarks points $\mathbf{X}_j \in \mathbb{R}^4$ (in homogenous coordinates) and m projection matrices $M_i \in \mathbb{R}^{3\times4}$ at which the m images were taken. These projection matrices are a composition $M = K[R|\mathbf{t}]$ of camera intrinsics K and camera extrinsics $(R, \mathbf{t})$. A 2D measurement (or observation) $x_{ij} \in \mathbb{R}^3$ in homogenous image coordinates is the projection of landmark $\mathbf{X}_j$ onto the image plane of the i-th image with camera matrix M_i.

Usually, SfM algorithms estimate preliminary reconstructions consisting of these entities, which are often not very accurate due to normally distributed measurement noise. To further refine these initial estimates of both scene structure and camera parameters, bundle adjustment is necessary as a final optimization step. As illustrated in figure 3.3, the term "bundle adjustment" originates from the light rays from the 3D landmarks into the optical centers of the cameras, which are adjusted to obtain an optimal result for both structure and camera parameters.

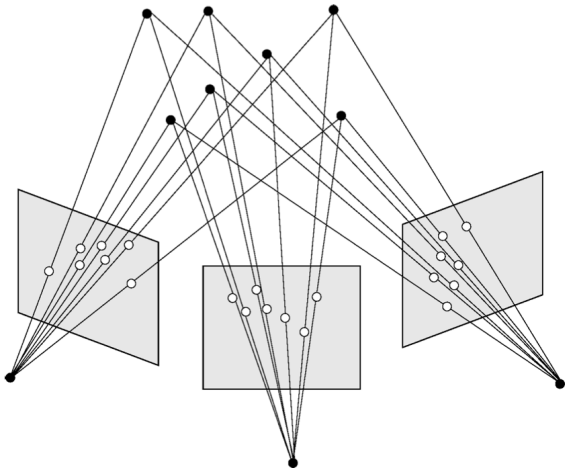


Figure 3.3: Bundle adjustment "adjusts" the light rays from the 3D landmarks into the optical centers of the cameras in order to refine the reconstructed map of a scene. Here, 7 points are projected into 3 images [28].

Formally, this is equivalent to the minimization of a cost function, which is defined as the 2D reprojection error between the observed and predicted image points:

$$\min_{M_i, \mathbf{X}_j} \sum_{j=1}^{n} \sum_{i=1}^{m} ||h(M_i, \mathbf{X}_j) - x_{ij}||^2. \tag{3.22}$$

The measurement function $h(M_i, \mathbf{X}_j)$ estimates the predicted projection of landmark j on image i and is typically non-linear. To solve this minimization, non-linear least-squares algorithms like Gauss-Newton or Levenberg-Marquardt (LM) have been applied successfully. Such methods typically work iteratively by first linearizing the problem around the current solution, solving it, and repeating these steps until convergence. In the following, the LM algorithm with its effective damping strategy, which allows fast convergence from a wide range of initial guesses, is explained in detail.

3.4.1 Least squares parameter estimation

Least Squares (LS) parameter estimation is an approach to provide an approximate solution for an overdetermined system of equations. Since there is usually no exact solution in the overdetermined case, LS minimization finds the solution closest to an exact solution by minimizing the sum of squared errors of the results of every equation. The most im-

portant application of LS is data fitting, when model parameters are determined based on uncertain and noisy observations.

With n independent data variables $x = (x_1, \ldots, x_n)^\top$ and dependent measurements $y = (y_1, \ldots, y_n)^\top$, the goal is to determine unknown model parameters $p = (p_1, \ldots, p_m)^\top$, which provide a best fit between model and measured data. A LS problem can then be formulated as the minimization of a squared distance objective function:

$$E_{LS} = \sum_{i=1}^{n} \|\mathbf{r}_i\|^2 = \sum_{i=1}^{n} \|f(\mathbf{x}_i; \mathbf{p}) - \mathbf{y}_i\|^2 . \tag{3.23}$$

The residuals $\mathbf{r}_i = f(\mathbf{x}_i; \mathbf{p}) - \mathbf{y}_i$ are the differences between the predictions $f(\mathbf{x}_i; \mathbf{p})$ and the corresponding measurements $\mathbf{y}_i$. By combining the residuals r_i into a vector $\epsilon = (r_1, \ldots, r_n)^\top$, this minimization problem can also be written as:

$$E_{LS} = \frac{1}{2}\epsilon^\top \epsilon, \tag{3.24}$$

with the constant $\frac{1}{2}$ introduced for later simplification.

Depending on the form of the function f, which in particular defines the way the residuals $\mathbf{r}_i$ are computed, it can be distinguished between linear and non-linear LS. In the following, we concentrate on the non-linear case with the problem of bundle adjustment in mind.

3.4.2 Levenberg-Marquardt algorithm

To solve the NLS optimization problem in bundle adjustment, we apply the iterative LM algorithm, which is basically a combination of Gauss Newton and gradient descent. Close to a local minimum, Gauss-Newton allows fast convergence. Far from the local minimum, the LM algorithm switches seamlessly to the slower gradient descent, which is guaranteed to converge.

As in the case of general LS, a parameter vector $p \in \mathbb{R}^m$ is mapped by a functional relation f to the predicted measurements $\hat{x} = f(p) \in \mathbb{R}^n$). With the measurement vector $x \in \mathbb{R}^n$, the residuals $\epsilon = x - \hat{x}$ and the initial parameter estimate p_0, the goal is to find the parameter vector $\hat{p}$ that minimizes the error $\epsilon^\top \epsilon$.

Under the assumption that f is locally linear, the initial parameter estimate p_0 is refined iteratively using incremental updates δ_p. For small $\|\delta_p\|$, an approximation of f can be obtained by $f(p + \delta_p) \approx f(p) + J\delta_p$ using the Jacobian $J = \partial f / \partial p$, which contains the corresponding gradient derivatives.

Since the LM algorithm is iterative, the parameter estimates (starting with p_0) are refined with each iteration until the final estimate $\hat{p}$ is found. This requires at each iteration to compute the step δ_p that minimizes:

$$\|x - f(p + \delta_p)\| \approx \|x - f(p) - J\delta_p\| = \|\epsilon - J\delta_p\| .$$

This can be achieved by solving the Linear Least Squares (LLS) problem $\|\epsilon - J\delta_p\|$ over δ_p using the normal equations

$$J^\top J\delta_p = J^\top \epsilon, \tag{3.25}$$

with the first order approximation $J^\top J$ to the Hessian of $\epsilon^\top \epsilon$ and the Gauss-Newton step δ_p. As the gradient of $\epsilon^\top \epsilon$ is $-J^\top \epsilon$, $J^\top \epsilon$ corresponds to the steepest descent direction.

In LM, an adaptive damping parameter $\lambda > 0$, which is adjusted from iteration to iteration, is additionally integrated into the normal equations from equation 3.25. The initial damping value is set to the product of a parameter τ (usually $\approx 10^{-2}$) with the maximum element of $J^\top J$ in the main diagonal. This yields the augmented normal equations:

$$(J^\top J + \lambda I)\delta_p = J^\top \epsilon. \tag{3.26}$$

If the value of δ_p obtained as solution of equation 3.26 reduces the error $\epsilon^\top \epsilon$, the increment is accepted and λ is decreased before the next iteration. In the case of an increased error through the obtained δ_p, the damping parameter λ is increased first and the augmented normal equations are solved again for δ_p until the determined δ_p leads to a decreased error. One iteration of the LM algorithm consequently consists of solving the augmented normal equations multiple times for different values of λ until an acceptable δ_p is obtained.

The behaviour of the LM algorithm is dependent on the adaptive damping parameter λ. For very small λ, the method behaves like Gauss-Newton. It then converges fast to the minimum in the case of an error function close to being quadratic in p. For large λ however, the normal equations are approximated by $\lambda\delta_p = J^\top \epsilon$. Since $J^\top \epsilon$ is the gradient vector of $\epsilon^\top \epsilon$, the direction of the parameter increment δ_p is equivalent to the method of gradient descent. With an increasing λ, the cost function $\epsilon^\top \epsilon$ decreases due to a decreasing length of the increment δ_p, which avoids excessively large Gauss-Newton steps.

All in all, the Gauss-Newton-like behaviour of the LM algorithm guarantees fast convergence close to the solution. The move to gradient descent can also provide a decrease in the cost function in problematic situations. By automatically adjusting the damping parameter, the LM algorithm is considered to be highly adaptive.

The LM algorithm finally terminates when at least one of the following criteria are true:

- the gradient magnitude is lower than a threshold.
- the relative magnitude of δ_p is lower than a threshold.
- the magnitude of the residual ϵ is lower than a threshold.
- the relative change of magnitude of the residual ϵ is lower than a threshold.
- the maximum number of iterations has been succeeded.

When a covariance matrix Σ_x, which describes the uncertainty of the measurements x, is available, it can be integrated into the previous algorithm by replacing the Euclidean norm of the error with the Σ_x^{-1}-norm (squared Mahalanobis distance) where applicable. The goal is then to minimize the squared Σ_x^{-1}-norm $\epsilon^\top \Sigma_x^{-1} \epsilon$, and the augmented normal equations become:

$$(J^\top \Sigma^{-1} J + \lambda I)\delta_p = J^\top \Sigma_x^{-1} \epsilon, \tag{3.27}$$

which is equivalent to solving a weighted least-squares problem.

3.4.3 Sparse bundle adjustment

After we have introduced the LM algorithm, its application in solving the optimization problem in bundle adjustment is explained. Hereby, an efficient sparse variant is introduced to exploit the sparsity pattern occurring in bundle adjustment, as 3D landmarks are usually not visible in all of the cameras.

As already described, n 3D landmarks are visible in m views with 2D projections x_{ij} of the j-th landmark on the i-th image. In bundle adjustment, we try to refine the initial estimates of camera and point parameters until we find the parameter set that optimizes the predicted 2D projections of the n points in the m views. In particular, we minimize the 2D reprojection error w.r.t. all 3D points and all cameras:

$$\min_{a_i,b_j} \sum_{j=1}^{n} \sum_{i=1}^{m} ||Q(a_i, b_j) - x_{ij}||^2. \tag{3.28}$$

With each camera parametrized by a vector a_i and each landmark parametrized by a vector b_j, $Q(a_i, b_j)$ predicts the projection of point j on image i. The advantage of this definition is that not all landmarks have to be visible in all views and that the minimization criterion of the 2D reprojection has a comprehensible physical meaning. Furthermore, this formulation is independent of the concrete parametrization of cameras and points. For dimensions α and β of each a_i and b_j respectively, the overall number of variables to be optimized is equal to $m\alpha + n\beta$, which can become large even in smaller bundle adjustment problems.

To integrate the bundle adjustment problem into the LM algorithm, we first combine the parameters of all cameras and all points into a single parameter vector $P \in \mathbb{R}^M$, such that

$$P = (a_1^\top, \ldots, a_m^\top, b_1^\top, \ldots, b_n^\top)^\top. \tag{3.29}$$

P_0 is the initial estimate of the parameter vector. The overall measurement vector $X \in \mathbb{R}^N$ consists of all the measured 2D image point coordinates of the 3D points in all the cameras, such that

$$X = (x_{11}^\top, \ldots, x_{1m}^\top, x_{21}^\top, \ldots, x_{2m}^\top, x_{n1}^\top, \ldots, x_{nm}^\top)^\top. \tag{3.30}$$

The uncertainty of the measurements X is expressed by the covariance matrix Σ_X. Because we lack further knowledge about the uncertainty, Σ_X is set to the identity matrix.

The functional relation f estimates a predicted measurement vector $\hat{X} = f(P)$, with

$$\hat{X} = (\hat{x}_{11}^\top, \ldots, \hat{x}_{1m}^\top, \hat{x}_{21}^\top, \ldots, \hat{x}_{2m}^\top, \hat{x}_{n1}^\top, \ldots, \hat{x}_{nm}^\top)^\top \quad \text{and} \quad \hat{x}_{ij} = Q(a_i, b_j). \tag{3.31}$$

With the residuals $\epsilon = X - \hat{X}$, the goal is to optimize the squared Mahalanobis distance $\epsilon^\top \Sigma_X^{-1} \epsilon$ over P. This optimization problem can be solved using the LM algorithm by repeatedly solving the augmented weighted normal equations:

$$(J^\top \Sigma_X^{-1} J + \lambda I)\delta = J^\top \Sigma_X^{-1} \epsilon. \tag{3.32}$$

Here, δ is the parameter update to be determined and J is the Jacobian of f.

Directly solving the normal equations using LM in its general form has cubic complexity in the number of parameters and hence represents a major computational bottleneck. In bundle adjustment, the normal equations of equation 3.32 exhibit a regular sparse block structure coming from the lack of interaction between different cameras and 3D points. This fact is exploited in sparse bundle adjustment, which takes advantage of the sparse structure of the problem to efficiently solve the normal equations.

Because image point $\hat{x}_{ij}$ depends only on the parameters of the i-th camera, $\partial \hat{x}_{ij}/\partial a_k = 0$ $(\forall i \neq k)$ holds. Accordingly, $\partial \hat{x}_{ij}/\partial b_k = 0$ $(\forall j \neq k)$ holds for the derivatives w.r.t. the parameters b_k of the k-th 3D point. We also define $A_{ij} = \partial \hat{x}_{ij}/\partial a_i$ and $B_{ij} = \partial \hat{x}_{ij}/\partial b_j$.

We can further divide the LM parameter update δ into structure and camera parameters $(\delta_a^\top, \delta_b^\top)^\top$, hence also $(\delta_{a_1}^\top, \ldots, \delta_{a_m}^\top, \delta_{b_1}^\top, \ldots, \delta_{b_n}^\top)^\top$. For a simple example of $m = 3$ cameras and $n = 4$ points, this leads to a Jacobian J of the following form, which can easily be extended to a larger number of cameras and points:

$$J = \frac{\partial X}{\partial P} = \begin{bmatrix} A_{11} & 0 & 0 & B_{11} & 0 & 0 & 0 \\ 0 & A_{12} & 0 & B_{12} & 0 & 0 & 0 \\ 0 & 0 & A_{13} & B_{13} & 0 & 0 & 0 \\ A_{11} & 0 & 0 & 0 & B_{21} & 0 & 0 \\ 0 & A_{12} & 0 & 0 & B_{22} & 0 & 0 \\ 0 & 0 & A_{13} & 0 & B_{23} & 0 & 0 \\ A_{11} & 0 & 0 & 0 & 0 & B_{31} & 0 \\ 0 & A_{12} & 0 & 0 & 0 & B_{32} & 0 \\ 0 & 0 & A_{13} & 0 & 0 & B_{33} & 0 \\ A_{11} & 0 & 0 & 0 & 0 & 0 & B_{41} \\ 0 & A_{12} & 0 & 0 & 0 & 0 & B_{42} \\ 0 & 0 & A_{13} & 0 & 0 & 0 & B_{43} \end{bmatrix} \tag{3.33}$$

To accommodate for the case that a point k is not visible in image l, both A_{kl} and B_{kl} are set to zero, such that these missing measurements have zero weight. As it can be evidently seen, the Jacobian J of equation 3.33 has a sparse block structure, which also makes the normal equations themselves sparse. To exploit this structure and avoid storing and operating on zero elements, both the left-hand side and right-hand side of the augmented normal equations in equation 3.32 are re-formulated in several steps. The full scheme for adjusting the augmented normal equations is explained in detail in [28].

Partitioning the parameters in sparse bundle adjustment into the separate sets of camera parameters and point parameters is important to reduce the computational complexity. As the dependency of the camera parameters on the point parameters is eliminated in the mentioned re-formulation, it is possible to first solve for the camera parameters update δ_a. This solution of a symmetric positive definite matrix can be performed efficiently using Cholesky factorization. Direct sparse Cholesky solvers can provide very efficient solutions even for large problems. Using the computed update of the camera parameters δ_a, the point parameter updates δ_b can subsequently be computed by back substitution. With usually significantly more points parameters than camera parameters in the optimization problem, it is reasonable to first solve for δ_a instead of first solving for δ_b.

By embedding the solution of the sparse normal equations based on Cholesky factorization into the LM algorithm, considerable computational savings can be achieved compared to solving the normal equations of equation 3.32 in each iteration.

4 3D Reconstruction

After the related work and the required preliminaries have been described in Chapter 3, the pipeline of the RGB-D based 3D reconstruction system for the workpiece reconstruction scenario is explained in detail in this chapter. A special emphasis is placed on the developed out-of-core bundle adjustment approach.

4.1 Basic approach

For the 3D workpiece reconstruction scenario a reconstruction pipeline similar to [16] has been developed. Figure 4.1 depicts a schematic overview of the different components.

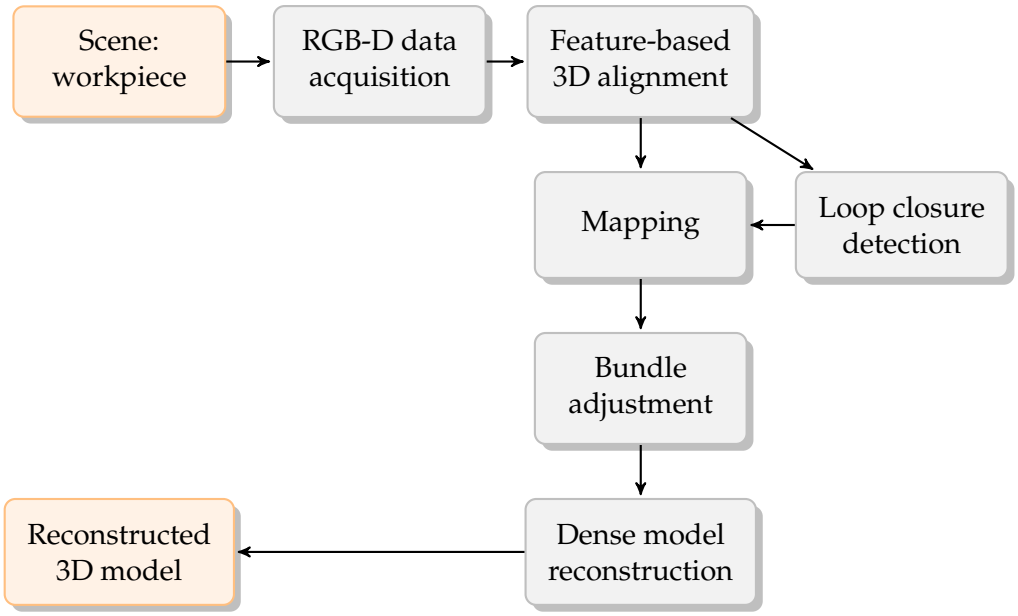


Figure 4.1: Overview of the 3D reconstruction framework developed within this thesis.

First, we acquire RGB-D data of the scene to be reconstructed using an RGB-D sensor. Detection and extraction of 2D visual features from the color images allow to match images pairwise. The determined 2D correspondences are augmented with depth information. The resulting 3D point correspondences are used for computing the relative 3D alignment between frames using a robust alignment method based on RANdom SAmple Consensus (RANSAC). Given these relative camera poses for frame pairs, the absolute pose of each camera frame w.r.t. the global world coordinate frame can be determined. Additionally, it is possible to detect loop closures in the camera trajectory when the images exhibit a scene that was already shown in a previous frame. Due to accumulated drift in

the absolute camera poses over time, bundle adjustment is integrated in order to refine the camera poses and 3D structure of the scene. Because of the large size of bundle adjustment problems in our workpiece reconstruction scenario, we develop and present an out-of-core bundle adjustment approach, which partitions the problem into smaller submaps. Finally, once the refined absolute camera poses of the frames are known, they can be utilized to integrate the frame data into a dense global 3D representation of the reconstructed model.

The details of the individual parts of the pipeline are explained in the following sections.

4.2 3D data acquisition

In each 3D reconstruction system, the first step is to acquire data of the scene of interest. As already mentioned, we use RGB-D sensors (such as the Microsoft Kinect) to obtain RGB-D data in order to reconstruct a 3D model of a scene.

In RGB-D data, each frame consists of a RGB color image and the corresponding depth map. In a color image, we can retrieve the observed color for a 2D point $(u, v)^\top$ using the following function:

$$I_{\text{RGB}} : \mathbb{R}^2 \rightarrow \mathbb{R}^3, (r, g, b)^\top = I_{\text{RGB}}(u, v). \tag{4.1}$$

A depth map is a 2D image, which contains depth values from the sensor to the next surface for each pixel of the color image. The corresponding depth for a 2D point is found through the lookup function:

$$I_d : \mathbb{R}^2 \rightarrow \mathbb{R}, z = I_d(u, v). \tag{4.2}$$

In the following, we assume that we have a one-to-one correspondence between pixels in the color image and their depth values, i.e. color and depth images are pre-registered.

Given the intrinsic parameters of the camera, each pixel can be reprojected into the local 3D camera coordinate frame using equation 3.13. This allows us to generate a full colored 3D point cloud based on the depth map.

4.2.1 RGB-D sensors

In principle, our approach is suitable for arbitrary sensors as long as they provide RGB-D data of a scene. Hence, it is also possible to integrate 3D scanners developed especially for specific application scenarios.

Sparse depth maps contain depth values for only few feature points and are consequently not sufficient for the reconstruction of dense, complete and accurate 3D models. It is consequently mandatory to acquire dense depth maps that contain depth values at each pixel for each frame. Depth maps can be computed from images of one or multiple standard CCD cameras using appropriate algorithms. While there has been significant progress in building dense depth maps from a single hand-held standard camera using multiple views in recent approaches [32, 34], the problem of determining the depth in real metric units still exists.

To gain the required dense depth maps, depth sensing devices like Time-of-flight cameras, 3D laser scanners or RGB-D cameras can be used. All of these devices yield metrically accurate depth values as discrete range measurements of the physical scene directly out of

the hardware. While the first two types of sensors are still quite expensive, the availability of the low-cost Microsoft Kinect RGB-D sensor has recently opened up the field of 3D reconstruction to a broader community. The Microsoft Kinect, which is used in [33], provides both RGB images along with dense depth maps at a resolution of 640×480 pixels at real-time frame rates of 30 Hz. In RGB-D sensors, structured light techniques [20] are utilized to generate dense depth maps in real-time. While the acquired depth data is of high quality, it is still affected by noise and missing depth values for some pixels. Furthermore, depth values are only determined up to a limited distance of typically less than 5 m.

In our work, we use an ASUS Xtion Pro Live [1] with an RGB-D sensor equivalent to the one of the Microsoft Kinect. To interface the RGB-D sensor, we employ the OpenNI frame-

Figure 4.2: ASUS Xtion Pro Live RGB-D sensor [1].

work [3]. RGB images and depth maps are already provided pre-registered by OpenNI to allow an easy one-to-one lookup of depth values for each pixel of an RGB image (and vice versa). While pre-computed intrinsic camera parameters (section 3.2) for these sensors are usually available, it is also possible to compute these parameters using a camera calibration framework like the *Camera calibration toolbox for MATLAB* [9].

4.2.2 Acquisition process and depth map processing

Because the accuracy of depth measurements of RGB-D sensors depends on the distance of the sensor to the measured surface, we use only depth measurements within a certain depth range for further processing. If a depth value of the acquired depth map exceeds a certain threshold value, the corresponding depth value is set to zero. This is equivalent to the fact that no depth measurement is available at this pixel. Discarding distant values with high uncertainty increases the accuracy of the subsequent 3D frame alignment and finally also increases the quality of the obtained reconstruction.

The actual process of acquiring data is performed by successively capturing frames of the workpiece while varying the camera pose w.r.t. the surface. In our scenario, we capture data with the RGB-D sensor by hand-held scanning. This setup is easy to accomplish and similar to many current research papers like [33]. We found it practical for most workpieces to acquire frames in two loops around the workpiece. The first loop captures the lower half of the object, while the second loop covers the upper half. For fully automated reconstruction scenarios, it is conceivable to mount the camera on a robot arm and move it according to preplanned trajectories. The main challenge when capturing the data is to keep the sensor at a reasonable distance to the surface of interest. If the sensor is too distant from the surface, only few details are visible in the captured color images. However, if the

sensor is too close to the object, then camera tracking might fail more easily, because depth sensors based on structured light have a minimum distance to determine depth values. To guarantee best quality for feature detection, 3D alignment and visual inspection, we decided to keep the sensor roughly at a distance between 0.70 m and 1.80 m to the surface (see figure 4.3).

Figure 4.3: RGB-D data of a workpiece is acquired by using a hand-held RGB-D sensor at a distance roughly between 0.70 m and 1.80 m to the surface.

After the acquisition of an RGB-D frame, the depth map of the scene is preprocessed in order to optimize the camera tracking quality. To suppress depth values with high uncertainty, pixel values in the depth map greater than a particular threshold (specific to the scene and application scenario) are rejected and set to zero. We additionally reduce noise in the depth map by applying a bilateral filter (as in [33]), which at the same time preserves discontinuities. Figure 4.4 illustrates the effects of the described preprocessing steps for the depth map of an input RGB-D frame.

4.3 Camera tracking using feature-based 3D alignment

One of the most important parts of every 3D reconstruction algorithm is the component for camera tracking, which determines the camera pose for each acquired RGB-D frame. The accuracy of the computed absolute camera poses in the global world coordinate frame is crucial for the quality of the reconstructed 3D model. Usually, camera tracking is performed by estimating the relative camera motions from frame to frame and then computing the absolute poses based on these relative poses.

There are different approaches to precisely determine the relative 3D pose between two subsequent RGB-D frames:

- The ICP algorithm [8] aligns two raw 3D point clouds based on an initial pose estimate by minimizing distances of point pairs of the two point clouds.
- Feature-based 3D alignment performs efficient camera tracking based on sparse visual features. Robust feature matching between two 2D color images (and projecting

Figure 4.4: RGB image (a) and depth map (b) of an RGB-D frame. The depth map is pre-processed using a bilateral filter (c), depth values exceeding a depth threshold are additionally cut off (d).

them back into 3D using their depth) results in 3D point correspondences. These correspondences are used to finally compute the 3D relative pose between these frames.

- Dense visual odometry approaches such as [48] and [41] are a current research topic of increasing interest. In contrast to feature-based 3D alignment, where a lot of information about the scene is discarded, all the color and 3D information available is utilized.

In order to choose the most suitable approach for this work, we examine the acquired RGB-D data in our workpiece reconstruction scenario. The workpieces to be reconstructed mostly exhibit very distinctive color features on the surface, while the 3D data in isolated form is often not very meaningful. These characteristics indicate that the ICP algorithm will be an inappropriate approach, because it does not utilize color information at all and might fail to work on only the 3D data available. Since dense visual odometry does not cope well with larger scale and orientation changes or occlusions so far, we consider sparse feature-based 3D alignment the most appropriate approach.

The chosen approach has the advantage that no initialization is required to determine the 3D alignment and is considered to be very general. Although we perform camera tracking based on sparse features only, dense 3D models can be generated by using the full dense point clouds of the determined camera poses. Given two RGB-D frames as described in section 4.2, the pairwise 3D alignment can be split up into multiple parts:

- Feature points are extracted from the two RGB images by searching for recognizable locations in the images based on their local appearance.
- Compact descriptors to represent these distinctive keypoints are extracted from the region around them.
- The detected features are matched across the two images in order to find the most likely feature point correspondences.
- The 2D feature correspondences are augmented with depth information in order to compute the relative 3D pose between the two frames. RANSAC is used to remove outliers (false matches) in the feature correspondences.

In the following, we explain the parts of our pairwise 3D alignment approach in more detail.

4.3.1 Feature detection

In order to perform the feature-based 3D alignment as presented above, it is crucial to effectively find visual features and describe them in an appropriate manner as a first step. Therefore, distinctive locations in the color images have to be found to provide points in the images that can most likely be detected in other images as well. For this purpose, proved approaches such as the Harris corner detector [21] can be applied. The keypoint detectors operate on grayscale images, which are produced from the RGB images in a straight-forward manner.

Once such keypoints have been found, it is important to extract robust descriptors that describe the local region around the keypoints in a stable manner. Since the local regions around the identical keypoint may vary from image to image, the descriptor has to provide invariance w.r.t. translation, rotation, scale and photometric changes. While preserving invariance to these kinds of changes, discriminability between different patches has still to be guaranteed. For this purpose, different types of descriptors have been developed and applied successfully in the past decades:

- Scale Invariant Feature Transform (SIFT) [29] is the probably most popular and successful type of feature detector. However, SIFT is computationally quite expensive and mostly not suitable for real-time applications.
- To compensate for this issue, a GPU-based implementation named SiftGPU [49] has been developed to speed up the computation of SIFT features.
- The Speeded-Up Robust Features (SURF) keypoint detector [7] is computationally less expensive than SIFT, but its use in real-time applications is still quite limited.
- With Oriented FAST and Rotated BRIEF (ORB) [39] a significantly faster alternative has been introduced recently. It was designed to give similar results as SIFT and SURF while at the same time being more efficient.

All of the feature detector types presented above have been integrated into our system. This allows us on the one hand to make the framework as flexible as possible and on the other hand to evaluate the suitability of the different approaches for our scenario (see chapter 5).

In practice, it is important to adjust the parameters for each feature type in such a way that the number of keypoints is reasonable. Too many features slow down the matching and possibly yield many false positives in the matching. Too few detected keypoints have the consequence, that the 3D frame alignment may fail due to too few keypoint correspondences. Figure 4.5 depicts an example image of one of the datasets acquired in this thesis, with keypoints detected by SiftGPU.

Figure 4.5: Example image of the WR/auger dataset with keypoints detected by SiftGPU.

Regardless of the concrete feature descriptor, each keypoint (u_k, v_k) in image i is described by a descriptor vector $\mathbf{d}_k^i$. These vectors consist of a certain number of elements (e.g. 128 elements in the case of SIFT), which can be compared with the descriptor vectors of other keypoints in the subsequent matching process. As final result of the feature detection component, we obtain a set of feature descriptors $D_i = \{\mathbf{d}_k^i\}$ from the color image of input frame i.

4.3.2 Feature matching

The detected feature descriptors can be utilized to establish correspondences between two images i and j. We assume, that the overlap between these images is sufficient to allow matching. Given this assumption, we further assume that features in the one image have their correspondence in the other images, although some for example might be occluded.

To find matches for a descriptor vector $\mathbf{d}_k^i$ in an image i, we have to compare it with the descriptors $\mathbf{d}_l^j$ in the second image j. In order to make the similarity between two vectors measurable, we have to apply a distance metric. In the case of SIFT and SURF features, the descriptor vectors can be compared using the Euclidean norm (l_2-norm). Given two n-dimensional descriptor vectors $\mathbf{d}_k^i$ and $\mathbf{d}_l^j$, their Euclidean norm is computed as

$$d(\mathbf{d}_k^i, \mathbf{d}_l^j) = \left\| \mathbf{d}_k^i - \mathbf{d}_l^j \right\|_2 = \sqrt{\sum_{s=1}^{n} (\mathbf{d}_{k_s}^i - \mathbf{d}_{l_s}^j)^2}, \tag{4.3}$$

where n is 128 in the case of SIFT. For efficiency reasons, ORB features are described by binary descriptors that allow a matching by their Hamming distance instead of their Euclidean norm. All in all, the smaller the distance $d(\mathbf{d}_k^i, \mathbf{d}_l^j)$ between two descriptor vectors is, the more probable it is that both descriptors describe the same feature.

With the distance metric defined as above, it is possible to search for correspondences in all the feature points of two different images. In a simple brute-force matching strategy, we compare each descriptor $\mathbf{d}_k^i$ of image i with all descriptors of image j and determine the closest descriptor $\mathbf{d}_l^j$. Figure 4.6 shows the feature correspondences found in the feature matching step for an image pair. However, this brute-force approach has quadratic

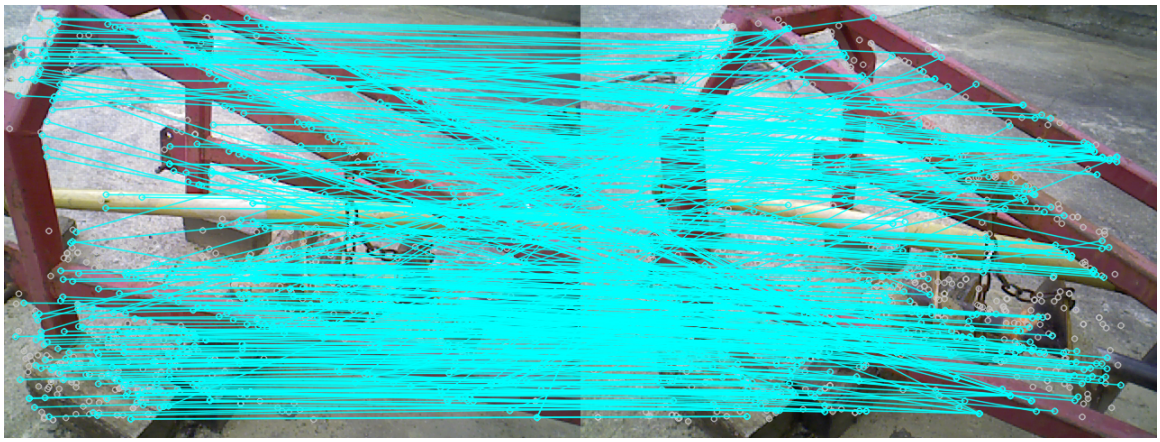


Figure 4.6: Corresponding features in two images after feature matching (including outliers).

complexity $O(n \cdot m)$, with n and m being the number of descriptor vectors in each image. For images with large numbers of feature vectors, the matching can be performed more efficiently by utilizing special indexing structures like multi-dimensional search trees. These nearest-neighbor-search methods allow searching for a given feature to rapidly search for near features and hence to find the best matches. As in the case of feature detection, we again rely on OpenCV [10] for feature matching. It provides a brute-force descriptor matcher as well as matcher based on nearest-neighbor-search on the basis of FLANN (Fast Library for Approximate Nearest Neighbors) [31].

After the feature matching step, we obtain pairs of corresponding feature descriptors $\{d_s^i\}$ and $\{d_s^j\}$. Here, corresponding descriptors across two images i and j are at the same

position s in their respective set. By looking up the respective keypoint $\mathbf{x}_s = (u_s, v_s)^\top \in \mathbb{R}^2$ for each descriptor $\{d_s\}$ in the images, we obtain pairs of corresponding 2D feature point sets:

$$K_i = \{\mathbf{x}_s^i\} \quad \text{and} \quad K_j = \{\mathbf{x}_s^j\}. \tag{4.4}$$

To constrain the computational requirements, only the best 512 matches (with smallest distances) per frame pair are processed further in the subsequent parts.

4.3.3 Robust 3D alignment using RANSAC

The sets of corresponding 2D feature points $K_i = \{\mathbf{x}_s^i\}$ and $K_j = \{\mathbf{x}_s^j\}$ obtained in the feature matching process can be utilized to estimate the relative pose $T_i^j \in SE(3)$ between the camera poses, at which the two frames i and j have been acquired.

Since we have the depth $d_s = I_d(u_s, v_s)$ for each keypoint $\mathbf{x}_s = (u_s, v_s)^\top$, we obtain the corresponding 3D coordinates for each 2D keypoint according to equation 3.13, such that $\mathbf{p}_s = \rho(u_s, v_s, d_s) \in \mathbb{R}^3$. This results in the two 3D point sets P_i and P_j:

$$P_i = \{\mathbf{p}_s^i\} \quad \text{and} \quad P_j = \{\mathbf{p}_s^j\}. \tag{4.5}$$

In theory, we can directly determine the relative 3D pose between the two frames using the SVD-based alignment presented in section 3.3 with these 3D point correspondences.

However, the putative matches obtained in the previous feature matching step still may contain many false positives, i.e. determined matches that in reality aren't keypoint correspondences. Wrong or missing depth values for keypoints also influence the quality of the computed 3D alignment. In the case of the used RGB-D sensors, the noise in the depth values is considered to be an issue, since the missing synchronization between color and infrared camera leads to inconsistencies of the depth image w.r.t. the color image.

The presented SVD-based approach for 3D alignment estimation based on point correspondences can only deal with Gaussian distributed noise in the measurements. To additionally cope with mismatches in the correspondences, which might make this least squares fit fail, a more robust approach is required as described in [22]. To get the true correspondences, we have to distinguish between inliers and outliers. This can be achieved by embedding the 3D transformation estimation of section 3.3 into the well-known approach of RANSAC [19]. This allows to robustify the estimation by coping with both noisy data and outliers.

The general objective of RANSAC is to perform a robust fit of a model to a dataset, to determine a set of inliers from the data and to compute the optimal model based only on these inliers. Applied to the 3D alignment estimation, we want to find the rigid body motion $C \in SE(3)$ that minimizes the sum of squared distances between the point correspondences of P_i and P_j, conforming to the condition that the distance between corresponding points is within a certain threshold. This problem consists of two parts: first the transformation is fitted to the data, then the correspondences are classified as inliers or outliers and the optimal rigid body motion is re-computed based on the computed inliers.

The application of the RANSAC algorithm on the 3D alignment estimation is outlined in algorithm 4.1.

Algorithm 4.1 Robust 3D alignment estimation using RANSAC.

Input: Corresponding 3D point sets $P_i = \{p_s^i\}$ and $P_j = \{p_s^j\}$ of size n with outliers; distance threshold t, inlier threshold T and maximum number of iterations N.

Objective: Estimate the optimal rigid body motion $T_i^j \in SE(3)$ between P_i and P_j and determine the final outlier-free point correspondences $\hat{P}_i$ and $\hat{P}_j$.

Algorithm:

(i) Initialize the sets of inliers: $\hat{P}_i = \varnothing$ and $\hat{P}_j = \varnothing$.

(ii) Randomly select a minimal sample set of four 3D point correspondences P_i^* and P_j^* from P_i and P_j.

(iii) Compute the initial transformation $C_0 \in SE(3)$ from P_i^* and P_j^* using the 3D alignment estimation method of section 3.3.

(iv) Determine the consensus set $\bar{P}_i = \{p_c^i\}$ and $\bar{P}_j = \{p_c^j\}$ of inliers, for which the following holds:

$$\left\| p_c^i - \mathcal{T}(C_0, p_c^j) \right\| <= t. \tag{4.6}$$

(v) If number of inliers $\left\|\bar{P}_i\right\| > \left\|\hat{P}_i\right\|$: set $\hat{P}_i = \bar{P}_i$ and $\hat{P}_j = \bar{P}_j$

(vi) If $\left\|\bar{P}_i\right\| > T$: estimate the transformation T_i^j from all correspondences in $\bar{P}_i$ and $\bar{P}_j$ and terminate the algorithm.

(vii) After N iterations: use the largest consensus set $(\hat{P}, \hat{Q})$ to re-estimate the 3D alignment T_i^j from $\hat{P}$ and $\hat{Q}$ and terminate the algorithm.

(viii) If $\left\|\bar{P}_i\right\| <= T$, go to step ii.

(ix) Optional: compute the set of inliers and estimate T_i^j. Repeat iteratively until the number of inliers converges.

In practice, the RANSAC parameters need to be adjusted according to the practical problem in advance to obtain the desired robust results:

- The distance threshold t should be set according to the measurement noise and is usually chosen empirically in practice. For the RGB-D data in our scenario, we found that a distance threshold of 0.02 m is reasonable for most of the cases.
- The number of iterations N should be chosen sufficiently high to ensure with probability p (usually 0.99), that at least one of the random samples of s points is free from outliers. An algorithm for an adaptive number of samples N is outlined in [22].
- The threshold T for the size of an acceptable consensus set should be set roughly similar to the believed number of inliers in a dataset: $T = (1 - \epsilon)n$ (with outlier probability ϵ).

After the RANSAC algorithm has terminated, we get the transformation T_i^j with the largest consensus set. By using the number of inliers as score, we continuously obtain better fits, which finally yield the best selected sample.

As we additionally get the sets of outlier-free 3D point correspondences $\hat{P}_i$ and $\hat{P}_j$, we consider only these inliers of 3D point correspondences for further processing. If the num-

ber of inliers is smaller than a user-defined threshold, the 3D alignment has failed. The reason for this is that the two frames show different scenes or have not enough overlap of the same scene. Figure 4.7 shows the keypoint correspondences from figure 4.6 after RANSAC-based outlier removal.

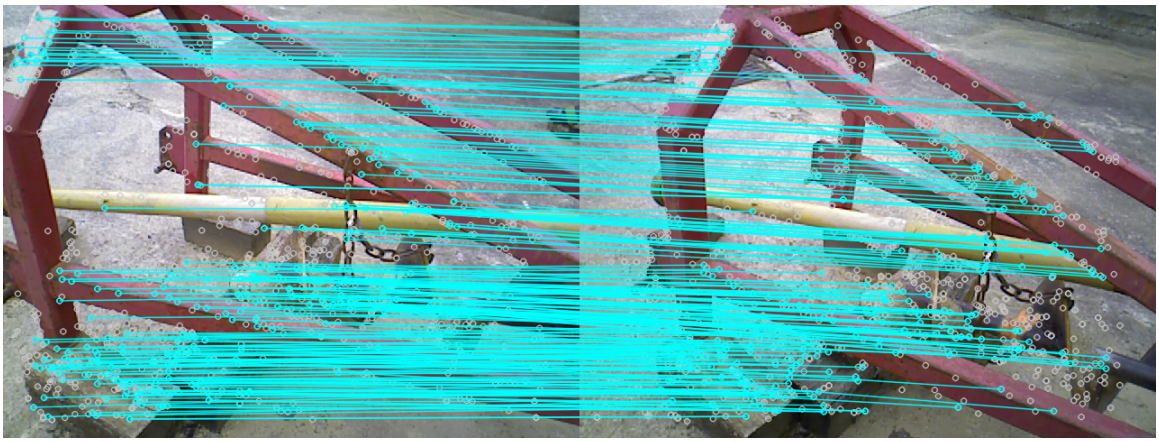


Figure 4.7: Feature correspondences of figure 4.6 after RANSAC-based outlier removal.

While RANSAC is a general approach, PROgressive SAmple Consensus (PROSAC) [11] is much faster for robust feature matching. This is accomplished by selecting samples from progressively increasing sets of top-ranked correspondences by exploiting the linear order of the set of correspondences based on a similarity function. However, we did not consider PROSAC in this thesis, since it does not affect the accuracy of the estimated transformation.

4.4 Mapping

SLAM systems usually maintain and update an internal 3D map of the environment. In our scenario, we use this 3D map mainly for camera tracking and for providing absolute initial estimates of camera poses and 3D landmarks in the global world coordinate frame. Afterwards, the final map is used as the input to a global optimization to further refine the contained camera poses, which will be described in section 4.5. In this section, the process of building the 3D map is described, which is continuously updated as more and more frames are integrated.

4.4.1 Graph-based map representation

The 3D map of the environment, which is maintained by our reconstruction framework, is usually represented as a graph in SLAM approaches. This so-called SLAM graph consists of:

- **Camera poses** $C = \{C_i \mid i \in 1 \dots M\}$, that describe the absolute camera poses $C_i \in SE(3)$ at which frame i was acquired. This rigid body motion transforms from the local camera coordinate frame into the global world coordinate frame.

- **3D landmarks** $X = \{\mathbf{X}_j \mid j \in 1 \dots N\}$, which represent the absolute global 3D location $\mathbf{X}_j = (x_j, y_j, z_j)^\top \in \mathbb{R}^3$ of distinctive keypoints seen in RGB-D frames.
- **Observations** $Z = \{\mathbf{z}_k \mid k \in 1 \dots K\}$, where $\mathbf{z}_k = (u_k, v_k, d_k)^\top = (u_j^i, v_j^i, d_j^i) \in \mathbb{R}^3$. These measurements are 2D image coordinates $(u_j^i, v_j^i)^\top$ and their respective depth value $d_j^i = I_d(u_j^i, v_j^i)$, under which landmark $\mathbf{X}_j$ is seen in frame i.

The camera poses and 3D landmarks constitute the two types of vertices of the graph. Poses are only connected directly to 3D landmarks, with the observations serving as graph edges. We denote the number of camera poses by M, the number of 3D landmarks by N and the number of observation over all frames by K. The notation used in this thesis is similar to [38]. Figure 4.8 depicts a simplified example of a SLAM graph.

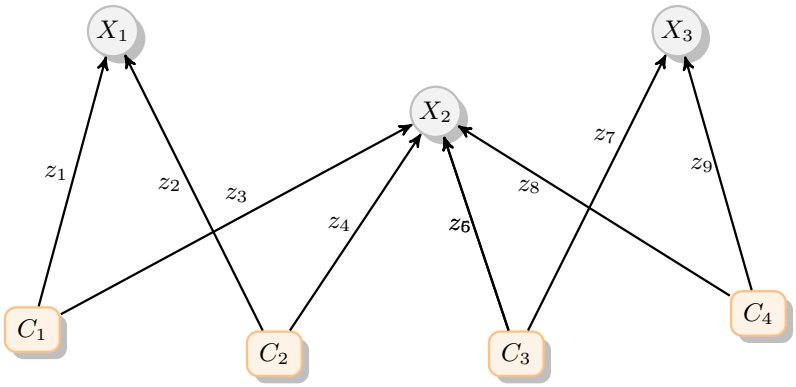


Figure 4.8: A 3D map represented as a graph. It consists of four camera poses C_i and three 3D landmarks $\mathbf{X}_j$, while the nine observations $\mathbf{z}_k$ serve as the edges of the graph.

4.4.2 Map update

The 3D map is constructed by continuously integrating new RGB-D input frames into the SLAM graph described in the previous section.

Based on the robust 3D alignment from 3D point correspondences, the relative pose for every pair of RGB-D frames can be determined. The estimated relative pose $T_{i-1}^i \in SE(3)$ between frames $i-1$ and i transforms from the previous coordinate frame $C_{i-1} \in SE(3)$ to the current coordinate system of frame i. Hence, it can be used to determine the absolute camera pose $C_i \in SE(3)$ of each new frame i based on the absolute pose of its immediate predecessor, which is also called frame-to-frame-tracking:

$$C_i = \mathfrak{T}(T_{i-1}^i, C_{i-1}) \tag{4.7}$$

This allows us to estimate initial guesses for all the absolute camera poses of the M RGB-D input frames. If the number of RANSAC inliers consistent to the estimated transformation is too small, then the frame alignment was not successful.

For all the measurements z_k, we can compute the absolute global estimate of the respective 3D landmark $\mathbf{X}_j$ visible in this measurement. We therefore project the observation

$\mathbf{z}_k = (u_j^i, v_j^i, d_j^i)^\top$ in camera frame i back into 3D and transform it into the global world coordinate frame:

$$\mathbf{X}_j = \mathcal{T}(C_i, \rho(u_j^i, v_j^i, d_j^i)). \tag{4.8}$$

By initializing the camera poses and 3D landmarks as described and connecting them with the observations, the map is constructed and continuously updated with new frames. The entire algorithm for integrating new frames with their camera poses, visible 3D landmarks and observations into the 3D map is outlined in algorithm 4.2.

Algorithm 4.2 Maintaining and updating the SLAM graph.

Input: M RGB-D frames with feature point observations $\mathbf{z}_k$ ($k \in 1 \dots K$) that describe the visibility of each 3D landmark X_j ($j \in 1 \dots N$) in the frames $i \in 1 \dots M$ (with poses $C_i \in SE(3)$).

Objective: Populate the SLAM graph (C, X, Z) and estimate the absolute camera poses C_i and 3D landmark locations X_j.

Algorithm:

(i) Initialize sets of camera poses $C = \varnothing$, landmarks $X = \varnothing$, observations $Z = \varnothing$.

(ii) Set camera pose of first frame to the origin of the world coordinate frame:

$$C_1 = (R_1 = I_{3\times 3}, \mathbf{t}_1 = (0,0,0)^\top), \quad C = \{C_1\}.$$

(iii) Acquire the next RGB-D frame: $i = i + 1$.

(iv) Estimate the relative pose $T_{i-1}^i \in SE(3)$ between the previous frame $i-1$ and the current frame i.

(v) If the alignment is not successful, discard the current frame i and go to step iii.

(vi) Initialize edges: $E = \{(i, i-1)\}$.

(vii) Compute the absolute pose C_i of frame i using equation 4.7.

(viii) Add current camera pose to the graph: $C = C \cup \{C_i\}$.

(ix) Check for loop closures with some frames l (see section 4.4.3); for every successful 3D alignment: $E = E \cup \{(i, l)\}$.

(x) For all pairs $(i, l) \in E$:

- Add all inlier measurements z_j^i and z_j^l of frames i and l, consistent to their common relative pose, to the graph: $Z = Z \cup \{z_j^i\} \cup \{z_j^l\}$.
- Add all 3D landmarks $\mathbf{X}_j$ visible in the measurements z_j^i and z_j^l to the graph: $X = X \cup \{\mathbf{X}_j\}$.

(xi) Stop if no more frames to be processed, otherwise back to step iii.

Although the robust 3D alignment estimation should result in quite accurate relative transformations, small errors in these relative transformations can accumulate to a visible drift in the absolute camera poses after a certain amount of frames.

4.4.3 Loop closure detection

The cumulative drift in the absolute poses mentioned in the previous section can deteriorate the overall quality of the map. This drift can nevertheless be reduced by optimizing the map using bundle adjustment, which is explained in depth in section 3.4. It is important for bundle adjustment to detect when the current frame shows roughly the same scene as in a previous frame, which may even have been acquired many frames before. This redundancy can be used for the optimization process. In order to refine the absolute camera poses, the relative poses between frames showing the same scene have to be determined. Therefore it is necessary to match and align the current frame not only with its immediate predecessor frame, but also with other previous frames. The process of finding these similar frames, which allow a successful 3D alignment with the current frame, is also called loop closure detection. Different approaches exist to detect loop closures efficiently.

A simple first approach for detecting loop closures is to apply a brute-force strategy, where the current frame is aligned with all previous frames. If an alignment is successful, both frames show the same scene and a loop closure is found. While this approach can yield the desired results, it is very inefficient, since the cost of detecting loop closures increases with each new frame due to the increasing number of previous frames to be matched.

For this reason, we select a subset of the previous frames and perform a 3D alignment only with the frames in this subset. By setting this subset to a constant size, we can detect loop closures for each new frame in constant time. In practice, we first add the three most recent frames to this subset. To detect loops in a larger scale, we additionally select 20 earlier frames, that we obtain through uniform sampling over all previous frames. Afterwards, we estimate the relative poses between the current frame and each of the frames in this subset in parallel. For each successfully aligned frame pair, we have detected a loop closure, so that the 3D map can be updated according to algorithm 4.2. Figure 4.9 shows the detected loop closures for the soil auger reconstruction (WR/auger) of chapter 5. For a better detection of loop closures, special approaches for appearance-based place recognition (e.g. FABMAP [12]) have been developed, which are however not within the scope of this thesis.

4.5 Out-of-core bundle adjustment

In the previous section, we have described how an initial map of the environment consisting of camera poses, 3D landmarks and measurements is represented and constructed. As we have seen, the absolute camera poses and the absolute location of the 3D landmarks w.r.t. the world coordinate system are determined by combining relative camera poses from frame to frame. However, while small errors in the relative pose estimates still allow for locally accurate estimates, they can lead to an accumulated drift in the absolute camera poses over time.

To account for this global misalignment and to guarantee a locally and globally correct reconstruction of the scene, sparse bundle adjustment (section 3.4.3) has been introduced as a global map optimization. While some SLAM frameworks perform pose-graph optimization based on the camera poses only, we integrate camera poses, 3D landmarks

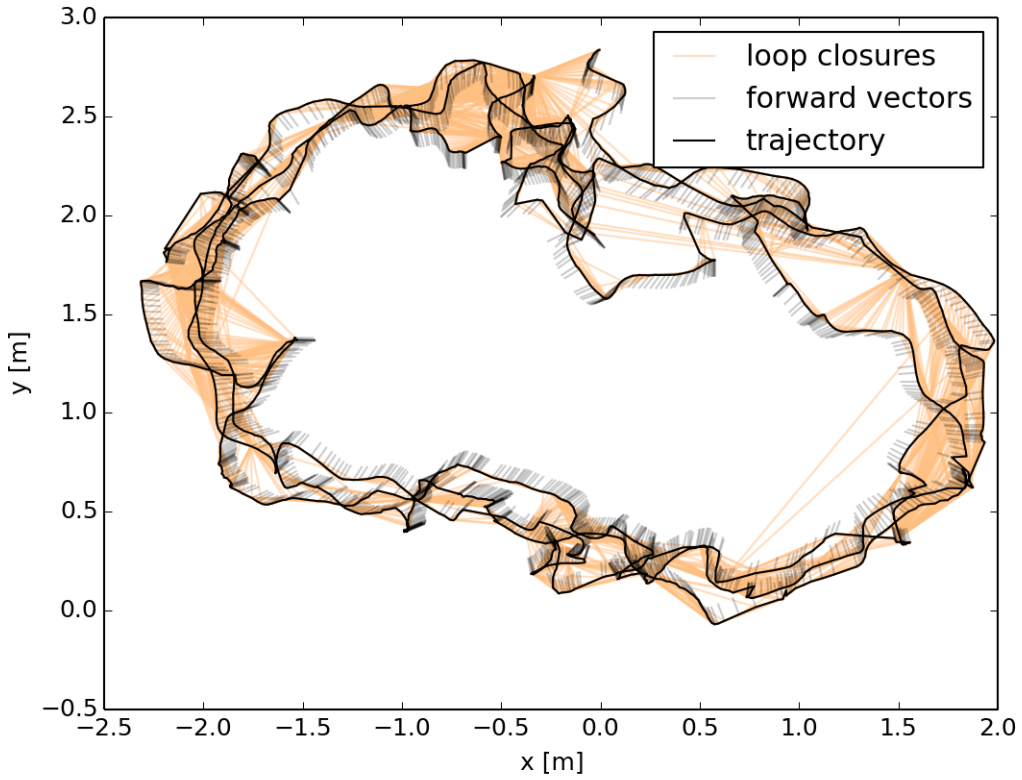


Figure 4.9: Estimated (unoptimized) camera trajectory with detected loop closures for the WR/auger dataset.

and their observations into the optimization in order to receive a reconstruction with best achievable accuracy. Due to the computational complexity and the high computational requirements of such full bundle adjustment approaches, these global optimizations are usually performed offline as a batch optimization after all input frames are integrated into the internal 3D map.

In the following, we use the notation that we introduced in section 4.4.1 for the entities of the SLAM graph. Hence, the optimization problem to be solved consists of M camera poses C_i, N 3D landmarks $\mathbf{X}_j$ and K observations $\mathbf{z}_k$. We assume the camera intrinsics f_x, f_y, c_x, c_y of the used RGB-D sensor to be known and constant for all frames of a dataset. The camera parameters C_i to be determined for each frame consist consequently only of rotational and translational components. To have a minimum number of parameters to be optimized, we represent the pose C_i with its twist coordinate representation $\xi_i = (\omega_1, \omega_2, \omega_3, v_1, v_2, v_3)^\top \in \mathbb{R}^6$. However, while this twist coordinate representation is important for the efficiency of solving the NLS problem, we use the abstract form $C_i \in SE(3)$ to denote the camera poses in the following.

To solve the NLS optimization problem, it is necessary to have good initializations for the optimization parameters to converge in an optimal global solution. The initial esti-

mates for the C_i and $\mathbf{X}_j$ in the map have been provided accordingly from frame-to-frame tracking as described in algorithm 4.2.

4.5.1 3D alignment error

Regular bundle adjustment refines both camera poses and 3D structure of the scene by minimizing the 2D reprojection errors. The 2D reprojection error is the difference between the actual 2D measurement $\bar{\mathbf{z}}_k = (u_k, v_k)^\top$ of a 3D landmark in an image and its predicted 2D position $\hat{\mathbf{z}}_k = h_k(C_{i_k}, \mathbf{X}_{j_k})$ based on the current estimates of the respective camera pose and landmark. Formally, bundle adjustment based on 2D reprojection errors tries to minimize

$$\sum_{k=1}^{K} ||\hat{\mathbf{z}}_k - \bar{\mathbf{z}}_k||^2. \tag{4.9}$$

The solution to this kind of NLS optimization problem is well-investigated and can be computed by applying the sparse LM algorithm presented in section 3.4.3.

However, the minimization of the 2D reprojection error does not utilize all the measured information available in this case. Due to the availability of RGB-D data, we can integrate depth measurements as additional constraints into bundle adjustment to improve robustness, convergence behaviour and accuracy. We generate a 3D measurement $\mathbf{Z}_k \in \mathbb{R}^3$ of a 3D landmark from its measurement $\mathbf{z}_k = (u_k, v_k, d_k)^\top$ by simply projecting it back into 3D according to equation 3.13:

$$\mathbf{Z}_k = \mathbf{Z}_{k_{ij}} = \rho(u_k, v_k, d_k). \tag{4.10}$$

The subscript ij in the notation $\mathbf{Z}_{k_{ij}}$ states explicitely, that the measurement relates camera pose i and landmark j. Instead of minimizing the 2D reprojection error, this allows us to directly minimize the 3D alignment error, which is defined as the difference between the measured and the predicted 3D position of a 3D landmark in the local camera coordinate frame.

With this definition of the 3D alignment error, the minimization problem in bundle adjustment can be re-formulated:

$$\sum_{k=1}^{K} ||\hat{\mathbf{Z}}_k - \mathbf{Z}_k||^2, \tag{4.11}$$

where $\hat{\mathbf{Z}}_k = h_k(C_{i_k}, \mathbf{X}_{j_k})$.

The prediction function h_k generates an artificial 3D measurement $\hat{\mathbf{Z}}_k$ of the j-th landmark $\mathbf{X}_{j_k}$ in the i-th frame with pose C_{i_k}. Concretely, the prediction is accomplished by simply transforming the landmark $\mathbf{X}_{j_k}$ in global coordinates back into the local camera coordinate frame using the respective absolute camera pose C_{i_k}:

$$h_k(C_{i_k}, \mathbf{X}_{j_k}) = \mathcal{T}^{-1}(C_{i_k}, \mathbf{X}_{j_k}). \tag{4.12}$$

It has to be noted that we do not focus on modeling the measurement uncertainties within the scope of this thesis. We therefore assume the covariance matrix to be the identity matrix for all measurements, why we left it out of the minimization problem formulations. However, when considering the uncertainties of these generated 3D measurements in the covariance matrix, we have to be aware of the fact, that the measured 2D feature location

and the depth measurement introduce different error types. This implies that a single scale factor for the covariance matrix is not sufficient, but we have to augment the covariance matrix by using one factor for x and y coordinates based on the certainty of feature detection and another factor for the depth measurement based on the RGB-D sensor specifics.

In practice, both problem formulations for optimizing the 2D reprojection error and the 3D alignment error can be fed directly into specialized libraries for solving bundle adjustment problems. Namely, there are the efficient state-of-the-art sparse graph optimizer *g2o* [27] and the NLS optimizer library *Ceres Solver* [5]. Due to first performance evaluations, we concentrated on *Ceres Solver* and used it for the practical solution of the presented optimization problems in this thesis.

4.5.2 Submap-based approach

We have already pointed out that solving bundle adjustment problems usually suffers from its high computational complexity and consequently becomes more and more problematic for an increasing amount of data.

With the amount of data occurring in the 3D workpiece reconstruction scenario, full bundle adjustment turns out to be inefficient as well as time- and memory-consuming. In order to make bundle adjustment more efficient and even feasible on machines with limited resources, alternative out-of-core approaches are required. Such out-of-core techniques pursue the goal to process only a small portion of a large dataset at a time and to obtain the overall result by finally combining the results from the processed subparts. This enables the processing of problems that cannot be kept in main memory at once, similar to bundle adjustment on machines with a limited amount of memory.

Accordingly, the submap-based bundle adjustment approaches presented in section 2.2.4 can be seen as out-of-core approaches. By partitioning the problem into several smaller submaps, it enables an independent optimization of the different submaps, such that not all submaps have to be held in physical memory at the same time. This allows even to efficiently obtain reconstructions that otherwise wouldn't fit into main memory.

The submap-based approach incorporated in this work is based on the work of Ni and Dellaert [37, 38] and similarly consists of the following four stages:

1. Partition the optimization graph into several submaps.
2. Optimize each submap internally.
3. Align the submaps globally.
4. Optimize each submap internally with fixed submap separators.

The individual stages of the developed submap-based bundle adjustment approach are presented in the following sections. We use the unoptimized SLAM graph depicted in figure 4.10 to illustrate the entities involved in each stage and to visualize the effects of the different stages.

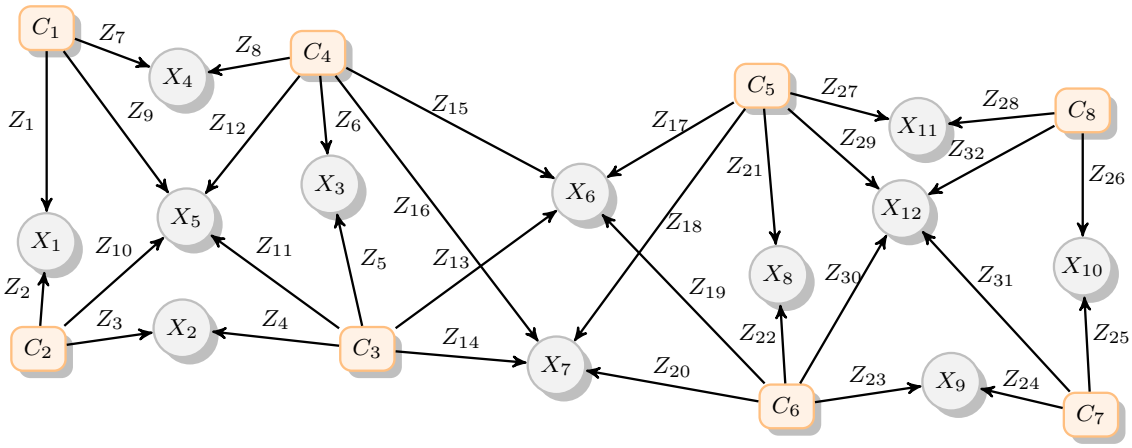


Figure 4.10: An unoptimized SLAM graph consisting of 8 camera poses C_i, 12 landmarks $\mathbf{X}_j$ and 32 observations $\mathbf{Z}_k$.

4.5.2.1 Graph partitioning into submaps

To facilitate a submap-based optimization approach, the full optimization graph has to be partitioned into a set of submaps in the first stage. We assume the number of submaps L to be given, whereas hints for finding an appropriate L are discussed in chapter 5.

When constructing a submap partitioning, it is of advantage to have all submaps of roughly equal size, such that the optimization efforts for the independent submaps are similar. We therefore partition the set of all camera poses C into L subsets $\tilde{C}_l$:

$$C = \{\tilde{C}_l \mid l \in 1 \dots L\} \tag{4.13}$$

of size $\tilde{M} = M/L$. Hence, submap l contains the camera poses from $(l-1)\tilde{M}+1$ to $l\tilde{M}$. While there are more advanced methods for graph partitioning, they are however not within the scope of this work. It is only mentioned that their goal is to find a graph cut that minimizes the number of measurements between the submaps to get optimal results in the global alignment stage.

Next, we assign a base node $B_l \in SE(3)$ to each submap as its local coordinate system. We then obtain a global set of base nodes:

$$B = \{B_l \mid l \in 1 \dots L\}. \tag{4.14}$$

We initialize the base node of a submap l with its first assigned camera pose:

$$B_l = C_{(l-1)\tilde{M}+1}. \tag{4.15}$$

When the submaps are populated, all camera poses and landmarks are parametrized w.r.t. the base node of the respective submap. We add all camera poses C_i to the set of camera poses of the respective submap $\tilde{C}_l$. Hereby, the poses are transformed into the submap's local coordinate system and thus expressed relative to the assigned base node:

$$\tilde{C}_l = \{\tilde{C}_i^l \mid i \in ((l-1)\tilde{M}+1) \dots (l\tilde{M})\}, \text{ with } \tilde{C}_i^l = \mathfrak{T}^{-1}(B_l, C_i). \tag{4.16}$$

For the first camera pose of each submap, the relative pose becomes equal to the identity transformation matrix, but it can change w.r.t. the base node in the later optimization due to its relative parametrization.

By iterating over all camera poses $\tilde{C}_i^l$ of a submap, we can add the connected measurements $\mathbf{Z}_{k_{ij}}$ to the set of submap measurements $\tilde{Z}_l$:

$$\tilde{Z}_l = \{\tilde{\mathbf{Z}}_{k_{ij}}^l \mid \tilde{C}_i^l \in \tilde{C}_l\}. \tag{4.17}$$

To populate the submap with landmarks, we iterate through all the measurements $\tilde{\mathbf{Z}}_{k_{ij}}^l$ of the submap and add the connected landmarks to the set of submap landmarks (expressed relative to the base node):

$$\tilde{X}_l = \{\tilde{\mathbf{X}}_j^l \mid \tilde{\mathbf{Z}}_{k_{ij}}^l \in \tilde{Z}_l\}, \text{ with } \tilde{\mathbf{X}}_j^l = \mathcal{T}^{-1}(B_l, \mathbf{X}_j). \tag{4.18}$$

If the landmark is seen by a camera pose contained in another submap, the landmark is added to the set of separator landmarks $\bar{X}_l$ of the submap:

$$\bar{X}_l = \{\mathbf{X}_j \mid \exists\tilde{\mathbf{Z}}_{k_{ij}}^l \in \tilde{Z}_l \wedge \exists\tilde{\mathbf{Z}}_{k_{\hat{i}j}}^{\hat{l}} \in \tilde{Z}_{\hat{l}} \wedge l \neq \hat{l}\}. \tag{4.19}$$

Using this method, separator landmarks are determined automatically. The set of separator landmarks of all submaps together is $\bar{\mathbf{X}} = \{\bar{\mathbf{X}}_l\}$. Measurements that connect a camera pose of a submap with a separator landmark are called inter-measurements, as they can be considered to be located between submaps.

After camera poses, landmarks and measurements have been added to the submap, the graph structure of each submap $\tilde{S}_l$ has been constructed, such that $\tilde{S}_l = (\tilde{C}_l, \tilde{X}_l, \tilde{Z}_l)$. To summarize, each camera pose is assigned to exactly one submap, while landmarks can be connected to camera poses of different submaps and can consequently be found in the landmark sets of multiple submaps. This relationship is essential for the global submap alignment, as the measurements between submaps can be exploited to optimize the global pose of the submaps and to refine the separator landmarks.

Figure 4.11 shows the graph of figure 4.10 partitioned into two submaps with attached base nodes. All camera poses and landmarks are expressed relative to the respective submap's base node.

In the case of limited main memory, we can store the camera poses, internal landmarks and measurements of each submap in separate files to implement an out-of-core technique. Another separate file contains the base nodes and the separator landmarks.

4.5.2.2 Submap optimization

After partitioning the full SLAM graph into submaps, we can optimize the submaps independently in the second stage to guarantee local accuracy. The processing in each submap $\tilde{S}_l$ can be done either completely in parallel, if enough resources are available, or in a regular consecutive manner.

The bundle adjustment problem in each submap consists of all its camera poses $\tilde{C}_l$, landmarks $\tilde{X}_l$ (both inner and separator landmarks) and measurements $\tilde{Z}_l$. Visually, all entities inside a submap's rectangle in figure 4.11 are used for optimizing the submap. Hence, we have to solve a few smaller optimization problems instead of one large problem. Using

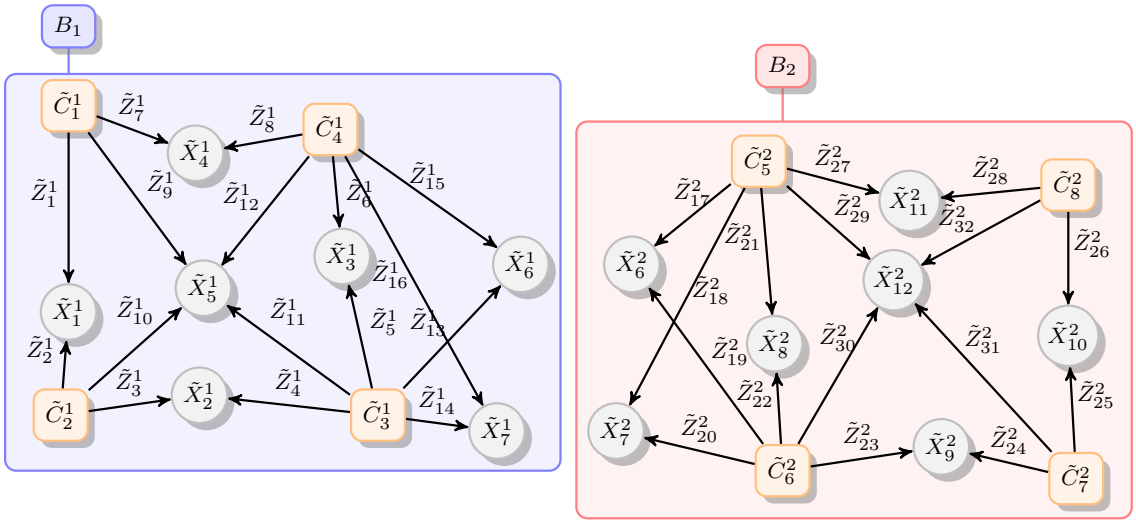


Figure 4.11: The map of figure 4.10 partitioned into two submaps. All camera poses and landmarks are expressed relative to their assigned base node, B_1 and B_2, respectively.

full bundle adjustment, the optimization of submap l itself is performed by minimizing the 3D alignment error as defined in equation 4.11:

$$\sum_{k=1}^{K_l} ||h_k(\tilde{C}_{i_k}^l, \tilde{X}_{j_k}^l) - \tilde{Z}_k^l||^2, \tag{4.20}$$

where K_l is the number of measurements in submap l.

After the variables $\tilde{C}_i^l$ and $\tilde{X}_j^l$ in all submaps $\tilde{S}_l$ have converged to their optimal values relative to their base node B_l, we have a metrically correct local reconstruction for each submap.

4.5.2.3 Global submaps alignment

To also obtain a correct global solution after the internal submap alignment, the global drift is eliminated by moving the base nodes and hence also the submaps. This is achieved by assembling the submaps and taking also the separator landmarks into account.

In this stage, the introduction of local coordinate systems in the submaps plays an important role. When the base nodes are moved in a global alignment step, the landmarks expressed relative to the base node move with the base node and hence keep their locally optimal values from the previous submap optimization.

In the global alignment stage, we perform a full optimization of a graph consisting only of the base nodes and the separator landmarks as vertices, connected by the relative locations of the separator landmarks in the submaps as measurements. We omit the internal landmarks and camera poses in the global alignment, as they have already been used to determine the optimal relative locations of the separator landmarks in each submap.

While the base nodes are already expressed in the global world coordinate system, we have to provide initial global estimates for the separator landmarks $\bar{\mathbf{X}}_j^l \in \bar{X}_l$. We therefore

estimate the global location of each landmark by transforming its relative location in the assigned submap with the respective base node's pose:

$$\bar{X}_l = \{\bar{\mathbf{X}}_j^l \mid \tilde{\mathbf{X}}_j^l \in \tilde{X}_l\}, \text{ with } \bar{\mathbf{X}}_j^l = \mathcal{T}(B_l, \tilde{\mathbf{X}}_j^l) \tag{4.21}$$

As each separator landmark is visible in multiple submaps, we initialize it only once with the location in the first submap it is visible in.

We make the approximation that we can use the location of the separator landmarks relative to the base node as measurements, because we assume their relative location to be optimal within the submap due to the performed local optimization. We compute these so-called inter-measurements as follows:

$$\bar{Z}_l = \{\bar{\mathbf{Z}}_k^l \mid \bar{\mathbf{Z}}_k^l = \tilde{\mathbf{X}}_k^l \wedge \tilde{\mathbf{X}}_k^l \in \bar{X}_l\}. \tag{4.22}$$

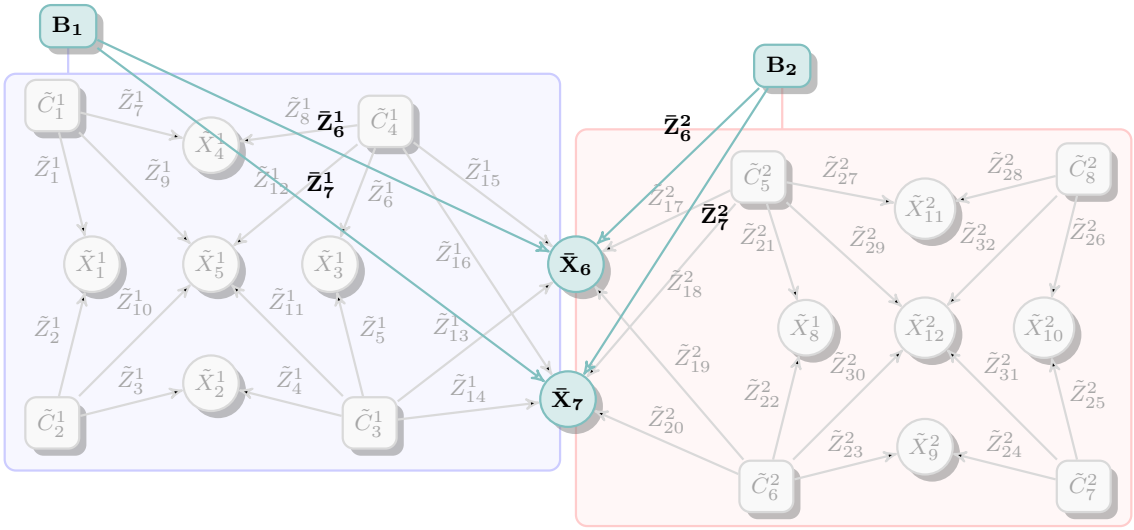


Figure 4.12: The global submaps alignment consists only of the base nodes and separator landmarks. The locations of the separator landmarks relative to their connected submaps' base nodes are used as measurements.

The actual global optimization consists of the entities highlighted in figure 4.12 and is performed by minimizing:

$$\sum_{k=1}^{\bar{K}} ||h_k(\tilde{B}_{l_k}, \bar{\mathbf{X}}_{j_k}) - \bar{\mathbf{Z}}_k^l||^2, \tag{4.23}$$

with $\bar{K}$ being the number of inter-measurements of all submaps.

With the optimization graph consisting only of base nodes and a limited number of separator landmarks, the global optimization itself can be computed very efficiently.

After this separator optimization, we obtain the optimal poses of the base nodes and the locations of the separator landmarks w.r.t. the global world coordinate system. By moving the base nodes, also the vertices contained in the respective submaps are moved, independent of how far the base nodes were moved. The effect of the base node movements is a

reduced global drift and the accomplishment of a globally metric reconstruction. While the camera poses and internal landmarks of the submaps still remain valid relative to the base nodes, the locations of the separator landmarks may have changed relative to the base nodes. This requires another final local alignment step in the submaps.

4.5.2.4 Internal submap update

To finally integrate the results of the changed separator landmark locations back into the submaps, we perform another internal submap optimization with fixed separator landmarks, i.e. the values of the separator landmarks are used in the optimization but they do not change any more. We therefore need to update the relative locations of the separator landmarks in each submap with the changed global locations:

$$\tilde{\mathbf{X}}_k^l = \mathcal{T}^{-1}(B_l, \bar{\mathbf{X}}_k^l). \tag{4.24}$$

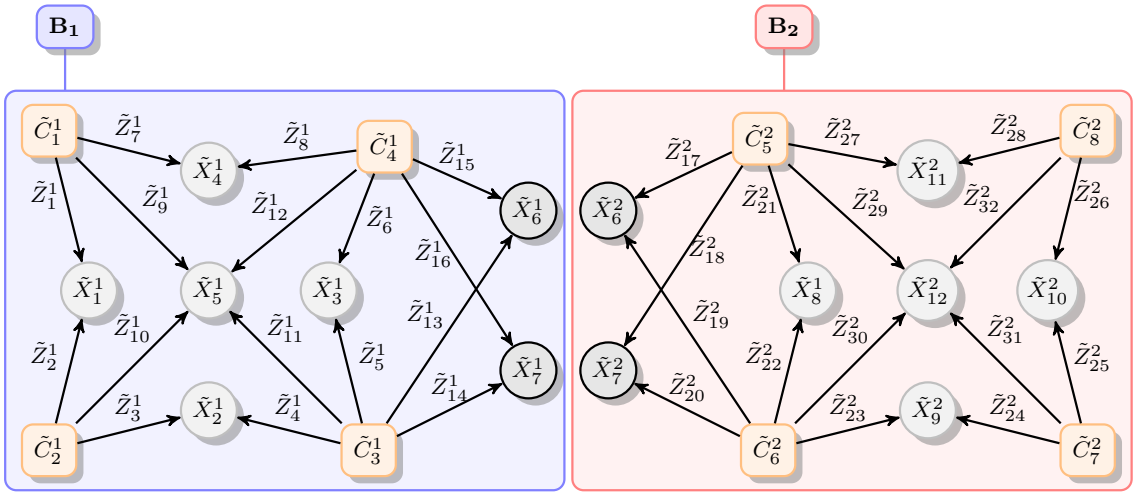


Figure 4.13: After the submaps and separator landmarks have been moved in the global alignment, we have to account for the changed separator landmarks. We therefore perform another internal optimization in each submap with fixed separator.

With these updated separator landmarks, we perform a full optimization with the same input as in the first stage. However, we set the relative locations of the separator landmarks fixed, such that the internal camera poses and landmarks are adjusted to optimally fit to the separator, while the separator landmarks stay in the same position. This usually also results in quick convergence and small movements of the internal poses and landmarks, since they were already optimized w.r.t. the base node's coordinate system.

Again, each submap (as depicted in figure 4.13) can be optimized independently by minimizing the bundle adjustment problem defined in equation 4.23.

After the submap optimizations have converged, the submaps' internal camera poses and landmarks may have changed again. To integrate these changes back into the separator and adjust the global submaps alignment, it is possible to iterate over the third and the

fourth stage of the algorithm until convergence. However, we found that the results did not change significantly by iterating over these two stages compared to performing global alignment and final internal submap optimization only once.

To get the final absolute results for the camera poses and landmarks in each submap, we need to transform them into the global coordinate frame again. Figure 4.14 shows the SLAM graph of figure 4.10 after we optimized it using our developed submap-based bundle adjustment approach.

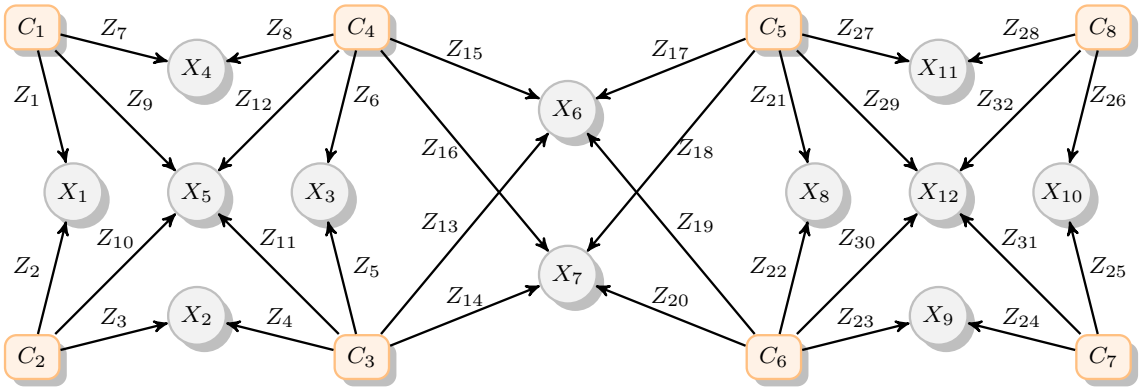


Figure 4.14: The final SLAM graph after we applied our submap-based bundle adjustment method. In contrast to its unoptimized state (see figure 4.10), we can easily identify the regular alignment of camera poses and landmarks in this figure.

The final optimal camera poses and 3D landmark locations are obtained by:

$$C_i = \mathfrak{T}(B_l, \tilde{C}_i^l) \quad \text{and} \quad X_j = \mathcal{T}(B_l, \tilde{\mathbf{X}}_j^l). \tag{4.25}$$

While we finally get both optimal camera poses and 3D landmarks, we only use the camera poses further for creating a dense model of the scene.

4.6 Dense reconstruction of 3D models

Once the absolute camera poses $C_i \in SE(3)$ for all input frames have been determined and refined, each acquired RGB-D frame can be integrated into a global 3D model representation. For this purpose, we first build a dense colored 3D point cloud of each RGB-D frame by re-projecting the pixels of the color image into the 3D camera coordinate frame using equation 3.13 based on their depth values. This point cloud can be integrated into the global model by transforming each contained point into the world coordinate frame using the absolute pose C_i. The full model of the scene is constructed by fusing the individual point clouds into a single representation that combines the information of all the frames.

The 3D representation for storing the model has to fulfill some requirements. First, it should be able to model arbitrary 3D scenes including the flexibility to dynamically expand the model map when needed. Another very important aspect is that the model must be storable efficiently in memory, even for an increasing number of integrated frames.

Moreover, it must always be possible to effortlessly integrate new frames, acquired at an arbitrary camera pose and showing a maybe unseen part of the scene, into the 3D model.

While there are various approaches for representing such models, e.g. TSDF volumes in [33] or a Surfel-based representation in [23], we focus on flexible point-based representations fulfilling the requirements stated above. In the following, the chosen 3D model representation, especially the way points are integrated and stored in the model, is discussed in detail.

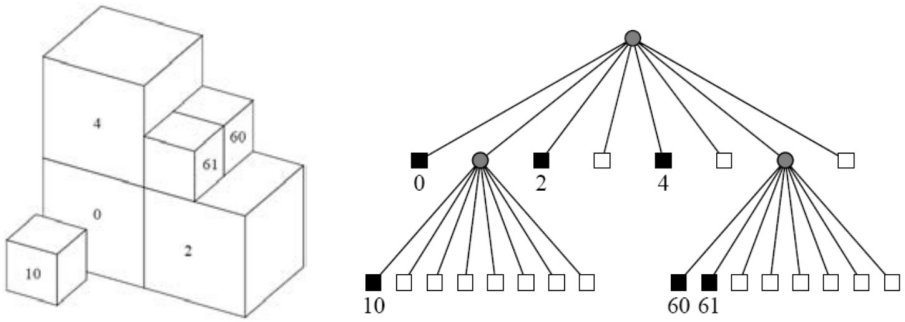
A simple approach for a global model representation is to add the colored 3D point clouds of all frames into a single **combined global point cloud**. Since no filtering is performed, the memory consumption of this representation increases with the number of integrated points. Consequently, there is no upper bound for the used memory and the processing requires vast computational resources.

In order to control the problematic memory consumption, a **volumetric grid** representation can be introduced, where the 3D space of the mapped model is discretized into cubic volumes of equal size (the so-called voxels). Each voxel contains the information if it is occupied and its respective point color. The dimensions of this voxel grid have to be predefined in advance, which results in a constant memory consumption, and needs to be large enough to contain all model data. The smaller the size of a voxel is defined, the more detailed the model can be stored, but also the more memory is required. Because the voxel grid cannot be extended during the reconstruction process and empty space is stored with as much memory as mapped voxels, we consider this approach to be very inflexible.

The inflexibility of volumetric grids w.r.t. the mapped scene can be compensated by managing the voxels in an efficient hierarchical tree structure. An **octree**, as described in [50] and [45], is such a tree-based representation for a 3D volume. We start with a single large cube that contains the modeled 3D space and represent the octree's root node. The octree is constructed by recursively subdividing this large cube into eight cubic subcubes until the voxel size reaches a given minimum value. For a higher resolution of the octree, the minimum voxel size can be decreased to obtain visually more detailed models, which also requires to process a higher amount of data. Since each node in a basic octree structure corresponds to a subvolume, a boolean property is used to model a node's occupancy state. In addition to the occupancy state, the RGB color of the voxel is stored for an occupied node. Figure 4.15a depicts an example octree that partitions the underlying 3D space. Its associated tree representation is illustrated in figure 4.15b.

When we add a point to the octree model representation, we first have to determine the existing octree subvolume it lies in by traversing the octree, starting from the root node. If this found subvolume is empty, then we mark it as occupied and additionally store the point's color in the voxel. However, if the determined voxel it lies in is already occupied, then we treat this leaf as inner node and subdivide it recursively until either both points do not lie in the same leaf voxel any more or we arrive at the minimum voxel size. If we arrive at the minimum voxel size, we mark the voxel as occupied and average the colors of the two points. In the other case of the points lying in different leaves, we mark the two leaf voxels as occupied and store the colors.

Octrees have the advantages that the mapped area can be extended dynamically and that



(a) (b)

Figure 4.15: (a) An example octree that partitions the underlying 3D space. (b) Its corresponding tree structure with interior nodes (gray), occupied (black) and unoccupied leaves (white). [45].

a high resolution at limited memory consumption is achievable. In contrast to volumetric grids, subvolumes don't have to be initialized before measurements are integrated.

In this thesis, it is assumed that sensor noise can be bounded by cutting off depth values with a distance too far from the sensor, since in Kinect-like sensors the uncertainty of measurements increases with distance. However, the octree-based approach stated above does not cope well with sensor noise and changing scenes, as invalid points can hardly be removed once they were integrated. Such scenarios require to model occupancy probabilistically and justify the use of probabilistic approaches like the octree-based mapping framework *OctoMap* [50]. In order to deal with scattered points introduced by sensor noise, we store for each octree voxel, in how many frames it was visible in. We consider leaf voxels seen in less than 5 frames as noise and hence remove them as a final post-processing step before generating the model.

After integrating all frames into the octree representation, we can finally generate a colored 3D point cloud, which consists of all occupied octree leaves, as output. This 3D model represents the final 3D reconstruction of the scene. Figure 4.16 illustrates the octree constructed for the downsampled 3D model of the WR/auger dataset.

Based on this representation, it is also possible to extract a triangular mesh of the scene, which is however not within the scope of this work.

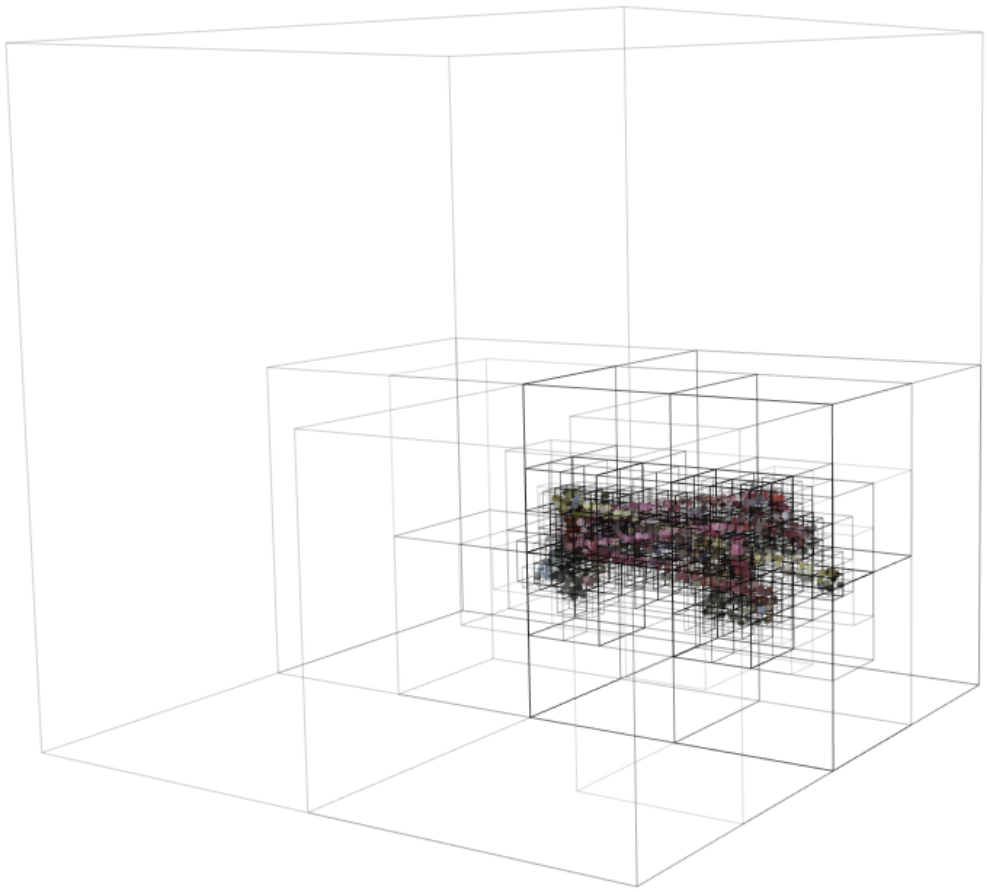


Figure 4.16: The reconstructed 3D model of the WR/auger dataset in an octree-based representation. The octree voxels are displayed in gray. For illustration purposes, the model was downsampled beforehand.

5 Evaluation and Experimental Results

This chapter provides an extensive evaluation of the developed 3D reconstruction system and the subsequent submap-based bundle adjustment described in the previous chapter. We present a detailed quantitative evaluation of both reconstruction algorithm and bundle adjustment approach using the TUM RGB-D benchmark by Sturm et al. [44, 43]. Further, we suggest a simple metric for finding a reasonable number of submaps for a given dataset. Afterwards, we present qualitative results of 3D reconstructions of several workpieces, which we obtained from applying the 3D reconstruction framework to acquired RGB-D datasets of workpieces.

All experiments were performed on a desktop PC with Intel Core i7-3770 CPU with 3.40GHz, 8GB RAM and an NVIDIA GeForce GTX Titan high-end graphics card.

5.1 Performance evaluation

To prove the suitability of the implemented reconstruction framework for the accurate reconstruction of workpieces, we first evaluate its performance in detail. By evaluating it with the mentioned TUM RGB-D benchmark, we obtain exact metric evidence for the achieved accuracy. Moreover, we show the flexibility of the developed approach, as it can not only reconstruct workpieces but also deal with arbitrary other RGB-D datasets.

An evaluation of the out-of-core bundle adjustment algorithm allows a comparison of full bundle adjustment with the submap-based approach w.r.t. accuracy and efficiency.

5.1.1 TUM RGB-D benchmark

The TUM RGB-D benchmark by Sturm et al. [44, 43] presents a complete benchmark targeted for the evaluation of RGB-D based visual SLAM systems. It consists of several RGB-D image sequences, which are recorded in different indoor environments and cover a variety of scenes and camera motions. This variety also guarantees a sufficient generalization of the evaluated SLAM approach.

For each sequence, both color and depth images are provided as well as the ground-truth camera poses determined by a high-accuracy motion-capture system. The availability of ground-truth data allows an objective and accurate comparison of different SLAM approaches, without the tedious task of acquiring ground-truth by oneself. Moreover, to ease the comparison of different SLAM approaches w.r.t. accuracy, a set of automatic evaluation tools is included.

Because of the advantages mentioned above, the TUM RGB-D benchmark serves as an optimal base for conducting a detailed quantitative performance evaluation of the 3D reconstruction framework and the submap-based bundle adjustment approach.

5.1.1.1 Datasets

For the evaluation of our approach, we selected a subset of the sequences provided in the TUM RGB-D benchmark. All datasets we acquired at a resolution of 640×480 at a frame rate of 30 Hz.

FR1/360: 360 degree turn in a typical office environment.
FR1/desk: Several sweeps over four desks in a typical office environment.
FR1/desk2: Second recording of the scene in FR1/desk.
FR1/plant: Plant filmed from 360 degrees.
FR1/room: Whole office scene with four desks (see FR1/desk); continues with outer walls of the room until loop closure.
FR1/rpy: Typical desk in an office environment; contains mostly RPY (roll-pitch-yaw) camera motions with fixed position.
FR1/teddy: Huge teddy bear filmed from different positions and orientations.
FR1/xyz: Typical desk in an office environment; only translatory (XYZ) camera motions with mostly fixed orientation.
FR2/desk: Typical office scene with two desks, computer monitor, keyboard, etc; trajectory contains a loop closure.
FR3/office: Household and office scene with much texture and structure; trajectory contains a large loop closure.

The characteristics of these sequences are presented in table 5.1.

Sequence	Duration	Duration with ground-truth	Ground-truth trajectory length	Trajectory dim.
FR1/360	28.69 s	28.70 s	5.8180 m	0.540 m x 0.460 m x 0.470 m
FR1/desk	23.40 s	23.35 s	9.2630 m	2.420 m x 1.340 m x 0.660 m
FR1/desk2	24.86 s	24.28 s	10.1610 m	2.440 m x 1.470 m x 0.520 m
FR1/plant	41.53 s	41.24 s	14.7950 m	1.710 m x 1.700 m x 1.070 m
FR1/room	48.90 s	48.87 s	15.9890 m	2.540 m x 2.210 m x 0.510 m
FR1/rpy	27.67 s	27.42 s	1.6640 m	0.150 m x 0.210 m x 0.210 m
FR1/teddy	50.82 s	50.78 s	15.7090 m	2.420 m x 2.240 m x 1.430 m
FR1/xyz	30.09 s	30.00 s	7.1120 m	0.460 m x 0.700 m x 0.440 m
FR2/desk	99.36 s	69.15 s	18.8800 m	3.900 m x 4.130 m x 0.570 m
FR3/office	87.09 s	87.10 s	21.4550 m	5.120 m x 4.890 m x 0.540 m

Table 5.1: Characteristics of the TUM RGB-D benchmark sequences.

5.1.1.2 Absolute trajectory error

The actual evaluation of a SLAM system in the TUM RGB-D benchmark is performed by measuring the Absolute Trajectory Error (ATE) between the estimated and the ground-truth camera trajectory. This evaluation metric allows us to determine the global consistency of the estimated trajectory by first aligning it with the ground-truth trajectory and then computing the absolute pose differences.

For computing the ATE, we compare a sequence of poses of the estimated camera trajectory $P_1, \ldots, P_n \in SE(3)$ with the poses of the ground-truth trajectory $Q_1, \ldots, Q_n \in SE(3)$.

We assume both trajectories to be time-synchronized, equally sampled and with the same lengths n. The poses of each trajectory are represented by homogenous transformation matrices, which are expressed relative to arbitrary global reference frames so that we have to align them first. This alignment is computed by finding the rigid body motion $S \in SE(3)$ between the trajectories using the method presented in section 3.3.

The ATE for the pose pair at time step i is defined as

$$F_i = Q_i^{-1} S P_i. \tag{5.1}$$

Finally, we can define the metric for comparing the estimated trajectory with the ground-truth. We therefore compute and evaluate the Root Mean Squared Error (RMSE) over all time indices by using only the translational components $\mathbf{t}_i \in \mathbb{R}^3$ of the poses $F_i = (R_i, \mathbf{t}_i)$:

$$\text{RMSE}(F_{1:n}) = \left(\frac{1}{n} \sum_{i=1}^{n} ||\mathbf{t}_i||^2 \right)^{1/2} \tag{5.2}$$

The TUM RGB-D benchmark provides tools to compute the ATE based on given ground-truth and estimated trajectories, as depicted in figure 5.1.

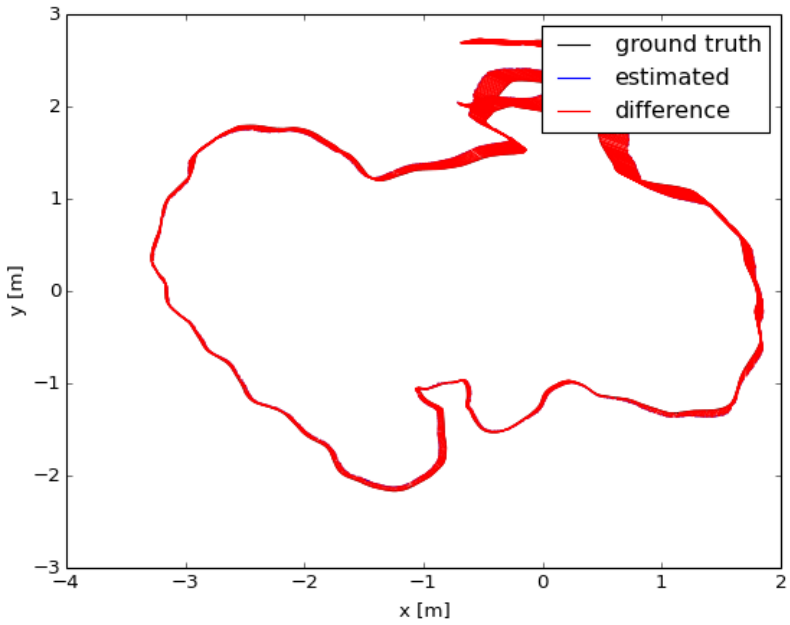


Figure 5.1: The root-mean square absolute trajectory error for the FR3/office sequence is the difference between the ground-truth and the estimated camera trajectory.

5.1.2 RGB-D-based 3D reconstruction system

In this section, we evaluate the accuracy and runtime of our developed SLAM framework using the selected sequences of the TUM RGB-D benchmark. Therefore, we consider the computed results of the 3D reconstruction system before the global bundle adjustment step.

First, we evaluate the accuracy and processing time of the different feature detectors SIFT, SiftGPU, SURF and ORB. We used the implementations provided by OpenCV [10] and the SiftGPU implementation of [49]. We adjusted the parameters of all detectors to get roughly the same amount of detected features per frame. Furthermore, we used only the 1024 best detected feature points in each frame. Table 5.2 shows the results of the different feature detectors.

Sequence	Frames	RMSE ATE [m]				Runtime (detection & extraction) [s]			
		ORB	SIFT	SiftGPU	SURF	ORB	SIFT	SiftGPU	SURF
FR1/360	744	0.242	0.245	0.108	0.186	0.0072	0.1076	0.0375	0.1437
FR1/desk	573	0.049	0.056	0.047	0.059	0.0096	0.1080	0.0336	0.1705
FR1/desk2	620	0.124	0.084	0.098	0.123	0.0090	0.1063	0.0373	0.1638
FR1/plant	1126	0.090	0.062	0.048	0.062	0.0091	0.0877	0.0354	0.1686
FR1/room	1352	0.248	0.302	0.275	0.219	0.0091	0.0943	0.0370	0.1759
FR1/rpy	694	0.051	0.049	0.046	0.055	0.0089	0.1016	0.0346	0.1397
FR1/teddy	1401	0.344	0.150	0.277	0.143	0.0091	0.0907	0.0406	0.1729
FR1/xyz	792	0.039	0.017	0.015	0.030	0.0098	0.1082	0.0333	0.1746
FR2/desk	2893	0.181	0.235	0.201	0.266	0.0097	0.0906	0.0376	0.1876
FR3/office	2488	0.215	0.248	0.176	0.476	0.0093	0.1174	0.0340	0.1785
Average		0.158	0.145	**0.129**	0.162	**0.0091**	0.1012	0.0361	0.1676

Table 5.2: Analysis of RMSE ATE and feature detection/extraction runtime of the different feature detectors.

While ORB is the fastest feature detector, it is accompanied by tracking failures and an increased drift. SiftGPU is slower than ORB, but still significantly faster than the other methods. By providing the best combination of speed and accuracy, SiftGPU clearly outperforms the other introduced feature detectors.

The feature matching methods explained in section 4.3.2 do not differ in their accuracy but only in their matching efficiency. Hence we do not provide an extensive evaluation between FLANN matching and the brute-force approach. However, we observed similar processing times (per frame pair) for the OpenCV implementations of both matching approaches based on features detected by SiftGPU. For this reason, we decided to use brute-force matching in the following.

Feature matching is required in order to estimate the relative poses between pairs of frames and thus determine the absolute camera pose of each added frame. To achieve more stable results for the initial camera poses and reduce global drift, each frame is matched against the three previous frames. We additionally match the current frame against a subset of 17 uniformly distributed frames to detect loop closures. To speed up the overall time of matching against predecessors, we performed frame matching and relative pose estimation in a parallelized manner. Hereby, each frame pair to be matched is processed in a separate thread.

To evaluate the overall runtime and processing frame rate of our developed system, the average runtimes of the individual processing steps per frame are listed in table 5.3.

Sequence	Pre-processing	Feature detection	Feature matching	Pose estimation	Time per frame
FR1/360	0.0179	0.0375	0.2769	0.1934	0.5258
FR1/desk	0.0329	0.0336	0.3488	0.1867	0.6020
FR1/desk2	0.0294	0.0373	0.306	0.1745	0.5473
FR1/plant	0.0334	0.0354	0.3233	0.1994	0.5914
FR1/room	0.0271	0.0370	0.3158	0.1873	0.5673
FR1/rpy	0.0181	0.0346	0.2736	0.1363	0.4627
FR1/teddy	0.0184	0.0406	0.3809	0.2191	0.6591
FR1/xyz	0.0184	0.0333	0.3586	0.2204	0.6307
FR2/desk	0.0230	0.0376	0.3695	0.2123	0.6424
FR3/office	0.0184	0.0340	0.2531	0.1898	0.4952
Average	0.0237	0.0361	0.3206	0.1919	**0.5724**

Table 5.3: Average runtimes per frame of the individual processing steps [s].

On average, our system required a processing time of 0.5724 s per frame, corresponding to a frame rate of roughly 2 Hz. Such frame rates are however not sufficient for real-time processing, but typical for approaches that rely on feature based frame alignment.

Summarized, we have developed a stable 3D reconstruction framework and we achieved good performance on most of the TUM RGB-D benchmark sequences even without global optimization.

5.1.3 Out-of-core bundle adjustment

Based on the evaluation of the 3D reconstruction framework in the previous section, we used SiftGPU and brute-force matching to build an initial sparse map for each dataset. For practical reasons, we stored these unoptimized results in files, so that we can use them to evaluate the different bundle adjustment methods with the same input data.

We utilized *Ceres Solver* to perform the actual NLS optimization and solve the bundle adjustment problem. An integration of the CXSparse library into Ceres Solver allows to exploit the sparseness pattern in bundle adjustment to achieve a major speed-up. CXSparse is an extension of CSparse [13] and provides an efficient sparse Cholesky decomposition.

In the following, we evaluate the proposed submap-based bundle adjustment algorithm by comparing it with full bundle adjustment based on the 3D alignment error. In particular, we measure and compare both its runtime and ATE relative to full bundle adjustment.

Both evaluations are performed for a varying number of submaps for each dataset. To compare the results of the different datasets, we express the number of submaps in a relative manner, concretely as the number of submaps per 100 frames computed as (absolute number of submaps L) / (absolute number of frames M). This allows us to determine the relative increase or decrease of runtime and metric accuracy depending on the number of submaps. Afterwards, we can suggest a metric for finding a good number of submaps, into which a dataset should be partitioned when our method is used.

5.1.3.1 Runtime

To evaluate the efficiency of our proposed method, we performed submap-based bundle adjustment with a varying number of submaps L for each dataset. The different submap

optimization runtimes t_{submap} are compared with the runtime of full 3D bundle adjustment t_{full} to get a percentage of runtime increase (positive sign) or decrease (negative sign). This relative runtime improvement (in %) is computed as $(t_{submap}/t_{full}) - 1$. The results are plotted in figure 5.2.

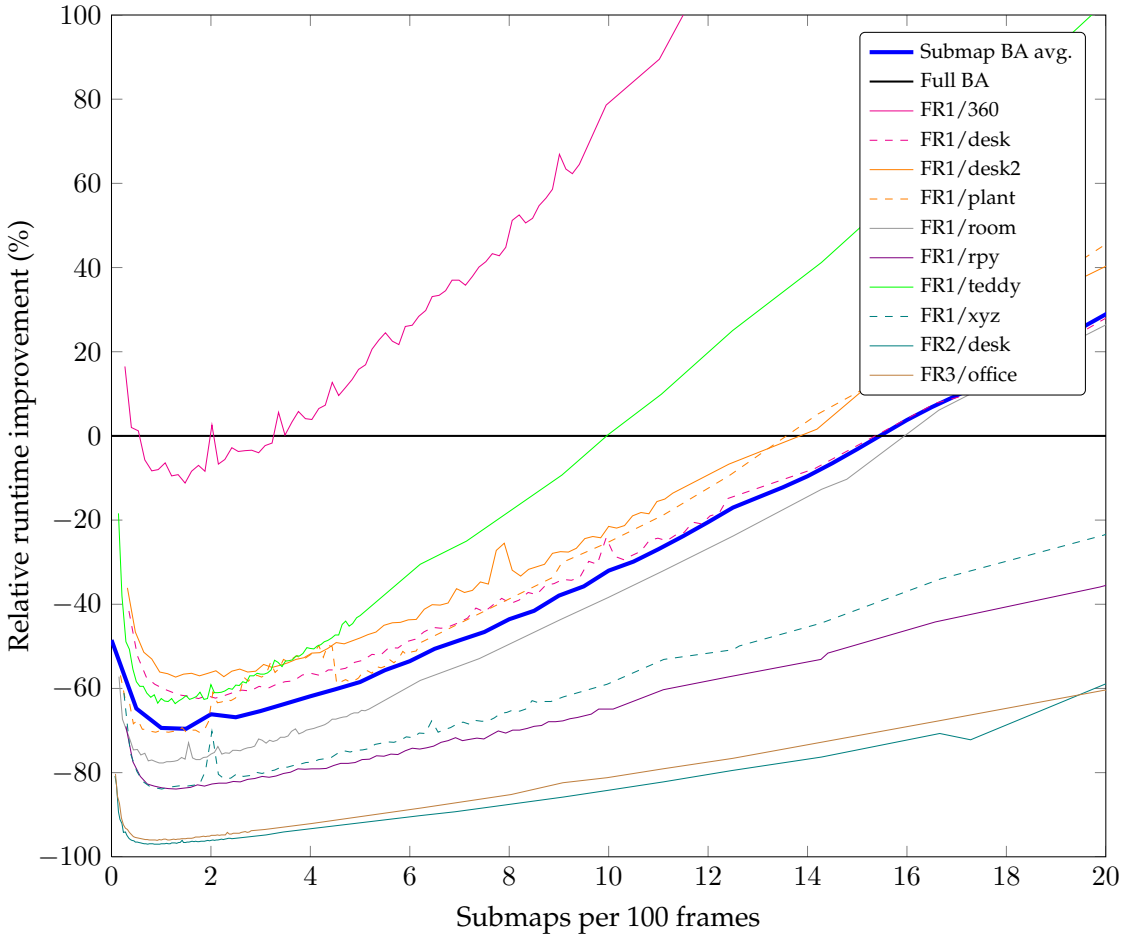


Figure 5.2: Runtime improvement (%) of submap-based bundle adjustment relative to full 3D bundle adjustment in dependency of the number of submaps per 100 frames. A reference line for full bundle adjustment (black) and the average of all datasets using our submap-based method (blue) are also shown.

We can derive from this plot, that the maximum speed-up in processing time is usually achieved at 0.5 submaps per 100 frames, i.e $L \sim 0.005M$. An improvement of roughly 40% can still be accomplished with 8 submaps per 100 frames (i.e. $L \sim 0.08M$) for almost all datasets. Moreover, we can see that the efficiency of bundle adjustment can be improved most for the long sequences FR3/office and FR2/desk, with an improvement up of more than 90% of the runtime of full bundle adjustment. For the sequences of small or medium size (with exception of FR1/360), an improvement between roughly 40% and 80% is possible depending on the dataset. For the FR1/360 sequence, full bundle adjustment converges already fast, probably due to initial estimates close to the optimal solution.

For a good efficiency of bundle adjustment, the number of submaps should be roughly in the range between 0.5 and 12 submaps per 100 frames Hence, the absolute number of submaps relative to the absolute number of frames M of a dataset should be within $0.005M \leq L \leq 0.12M$. However, for larger sequences a larger value for the upper bound seems to be valid as well.

5.1.3.2 Accuracy

Similar to the runtime evaluation, we also evaluate the accuracy of our method for a varying number of submaps for each dataset.

The accuracies ATE_{submap} of the submap optimization runs were compared with the accuracy of full 3D bundle adjustment ATE_{full}. The relative improvement is computed using the formula $(ATE_{submap}/ATE_{full}) - 1$. This results in a percentage increase (positive sign) or decrease (negative sign) of the ATE, which is presented in the plot of figure 5.3.

We can deduce from this plot, that we approach the accuracy of full bundle adjustment with an increasing number of submaps. The reason for this is, that the number of base nodes involved in the global alignment increases and thus the global drift can be reduced.

Further, we find a somehow oszillating behaviour of the ATE for a small numbers of submaps. With only few submaps, a good global alignment may not be achievable, because complex camera trajectories cannot be compensated by moving only few base nodes. The oszillations come from different submap partitionings for a different numbers of submaps, such that favorable cuts between submaps lead to better global submap alignments. However, favorable submap partitionings strongly depend on the characteristics of each specific SLAM graph and could be reduced by finding better graph cuts, which are however not within the scope of this thesis.

It has to be mentioned, that our submap-based approach achieves better results for some datasets than full bundle adjustment. While full bundle adjustment often converges not in a global, but in a local optimum due to bad initial estimates for the camera poses and landmarks, our approach can overcome these bad initializations due to the robust global alignment based on the preceding internal submap optimizations.

Since we want to achieve a good global alignment, we have to avoid the consequences of unfavorable submap partitioning. We assure this with a number of submaps that leads to a better global alignment and a reduced global drift. Hence, we propose a number of at least 8 submaps per 100 frames to obtain stable results, i.e. $L \geq 0.08M$ for the absolute number of submaps.

5.1.3.3 Absolute results and comparison

After we have evaluated runtime and accuracy of the developed submap-based bundle adjustment method in dependency of the number of submaps, we can suggest a simple metric for finding a reasonable number of submaps for optimizing a dataset. Afterwards, we give a summarized comparison of the different bundle adjustment approaches and compare our results with the RGB-D SLAM system.

First, to find a reasonable number of submaps, we have to combine the suggestions for the optimal number of submaps L made in the previous sections 5.1.3.1 and 5.1.3.2. While

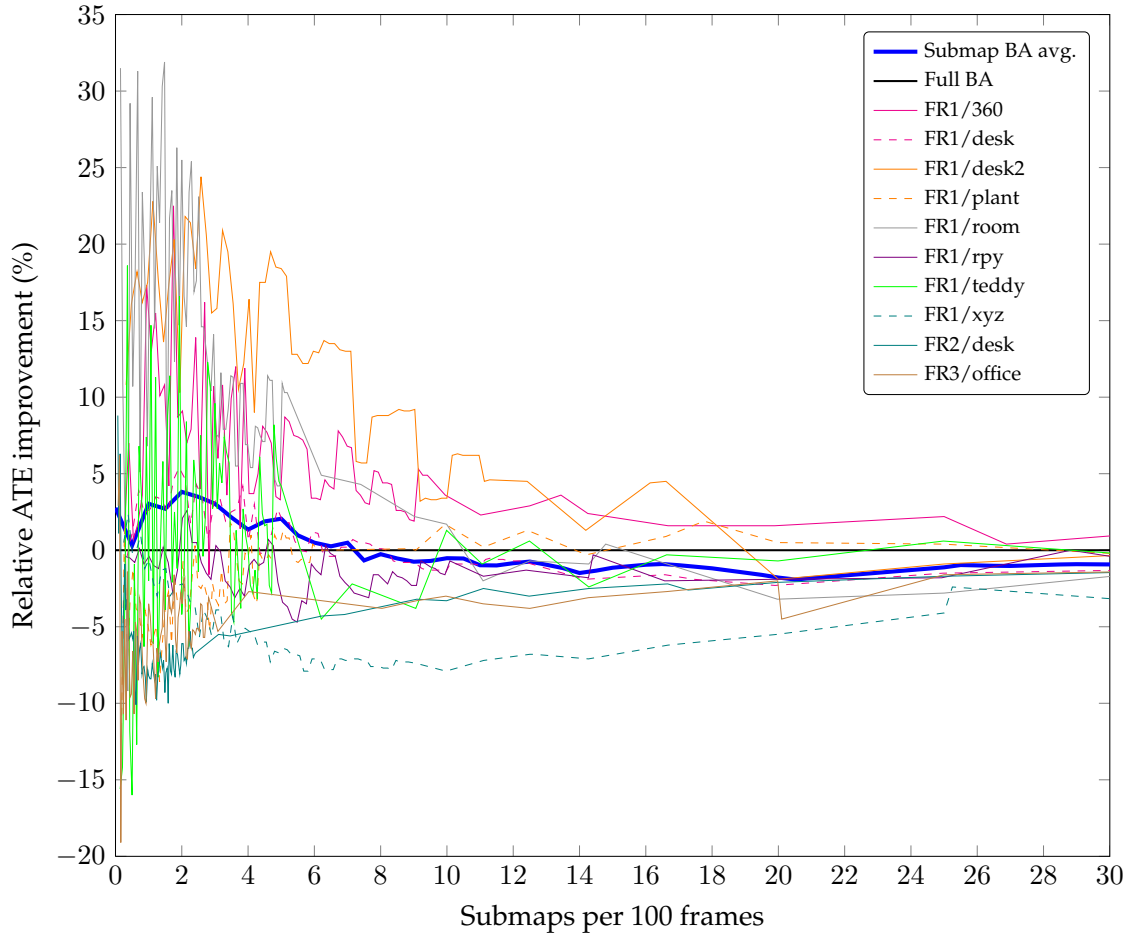


Figure 5.3: ATE improvement (%) of submap-based bundle adjustment relative to full 3D bundle adjustment in dependency of the number of submaps per 100 frames. A reference line for full bundle adjustment (black) and the average of all datasets using our submap-based method (blue) are also shown.

bundle adjustment is very efficient for a small number of submaps (e.g. $L \sim 0.005M$), we cannot make certain assumptions about the corresponding ATE due to its oszillating behaviour. For a larger number of submaps (e.g. $L \geq 0.20M$), we obtain stable results for the ATE comparable to full bundle adjustment, which however come along with a lack of efficiency for smaller datasets. To get a good tradeoff between efficiency and accuracy, we found a number of 10 submaps per 100 frames, i.e. $L \sim 0.10M$, to be reasonable for most of the datasets. For example, a dataset consisting of 500 frames should be partitioned into $L \sim 0.10 \cdot 500 = 50$ submaps.

To finally obtain absolute results for our method, we use the reconstructed maps computed by our developed 3D reconstruction system and optimize them with the different bundle adjustment methods. We performed full bundle adjustment using the 2D reprojection error as well as using the 3D alignment error. Moreover, we computed the number of

submaps for each dataset according to the suggestion formulated above ($L \sim 0.10M$) and performed the proposed submap-based bundle adjustment. Table 5.4 shows the absolute results of the different approaches for the TUM RGB-D benchmark sequences. To demonstrate, that the performance of our submap-based approach is comparable to the state of the art, we also compare it with the results of RGB-D SLAM.

Sequence	No BA	Full 2D	Full 3D		Submap-based					RGB-D SLAM
	ATE	ATE	ATE	time	submaps	ATE	±(%)	time	±(%)	ATE
FR1/360	0.108	0.099	0.077	12.66	74	0.079	+3.6	22.62	+78.6	0.079
FR1/desk	0.047	0.021	0.022	28.97	57	0.022	-1.5	21.96	-24.2	0.023
FR1/desk2	0.098	0.044	0.030	27.23	62	0.031	+3.4	21.36	-21.5	0.043
FR1/plant	0.048	0.023	0.042	66.27	112	0.043	+1.7	49.36	-25.5	0.091
FR1/room	0.275	0.228	0.085	125.46	135	0.086	+1.7	77.30	-38.4	0.084
FR1/rpy	0.046	0.058	0.027	67.56	69	0.027	-1.6	23.69	-64.9	0.026
FR1/teddy	0.277	0.060	0.056	67.88	140	0.057	+1.3	68.06	+0.3	0.076
FR1/xyz	0.015	0.013	0.013	96.87	79	0.013	-7.9	39.72	-59.0	0.014
FR2/desk	0.201	0.080	0.079	2355.26	289	0.076	-3.3	372.20	-84.2	-
FR3/office	0.176	0.039	0.036	1290.24	248	0.035	-3.0	242.88	-81.2	-
average	0.129	0.066	0.047			0.047	-0.5		-32.0	0.054

Table 5.4: RMSE ATE [m] and runtimes [s] for the TUM RGB-D benchmark sequences. The absolute results of submap-based bundle adjustment are presented, compared relative to full 3D bundle adjustment and compared with RGB-D SLAM.

The results in the table show, that our submap-based bundle adjustment method (with $L \sim 0.10M$) achieves an average ATE of 0.047 m over all sequences. While this accuracy is similar to the results of full 3D bundle adjustment, its average runtime is even 32.0% faster. It is also noticeable that our proposed approach works especially well for the larger sequences FR3/office and FR2/desk, for which an average speed-up of 82.7% is achieved. However, the outlier FR1/360 shows that for bundle adjustment problems, which are initialized with estimates close to the optimum, full bundle adjustment can converge fast. In these cases, submap-based approaches do not lead to improvements in efficiency, but have contrary effects.

Compared to the RGB-D SLAM system, we achieve a higher accuracy on average. This is due to the fact that RGB-D SLAM performs a simplified pose-graph optimization, which on the other hand makes it more efficient than our framework.

5.2 Results of 3D workpiece reconstruction

In the previous section, we have performed an extensive evaluation of the 3D reconstruction system and the submap-based bundle adjustment approach using ground-truth data provided by the TUM RGB-D benchmark.

Since we have demonstrated the validity of our framework, we can use it without further restrictions to create 3D reconstructions of workpieces. In the following, we illustrate the achieved results for three different workpieces:

WR/auger: a soil auger used to dig holes into the soil.
WR/lawn: a lawn tractor used to mow larger lawns.
WR/tractor: a small farm tractor used for mostly agricultural purposes.

All sequences were acquired with a hand-held ASUS Xtion Pro Live RGB-D sensor at a resolution of 640×480, as described in section 4.2.1. The capture rate was only 15 Hz due to the hardware limitations of the laptop used to acquire the datasets.

In the following sections, we introduce the acquired datasets for the workpieces and depict subsets of their input frames. We also show some views of the reconstructed 3D models as qualitative visual results. To perform measurements on these digital models in practice, special 3D viewer software like MeshLab [2] can be used.

All in all, the resulting reconstructions are highly detailed and metrically accurate, so that they can be used in practice for measuring tasks or for visual inspection without physical access to the workpiece.

5.2.1 Soil auger

The WR/auger dataset contains 2349 frames of a soil auger that can be mounted onto a farm tractor (e.g. the one from the WR/tractor dataset). The auger's dimensions are 0.85 m x 1.16 m x 2.80 m (width x height x length). It consists of three major parts, as it can be seen in the images in figure 5.4: the red frame, the actual auger of twisted shape and a yellow power take-off (PTO) shaft, which can be attached to a tractor.

The map produced by the 3D reconstruction system contains 2349 camera poses, 156974 3D landmarks and 1086734 observations. We optimized the map with our submap-based bundle adjustment method with 230 submaps in 195 s. The difference between the estimated camera trajectory before and after bundle adjustment is depicted in figure 5.5. We acquired frames in three loops around the object, resulting in a final optimized trajectory of length 52.88 m.

The obtained reconstruction is illustrated in figures 5.6 and 5.7. We can see that the achieved high visual quality is sufficient to inspect the surface and detect dents and deformations, which can occur when the auger collides with unexpected hard objects in the soil. The high quality is supported by the fact that no drift is visible in the final reconstructed model. This makes the reconstruction also suitable for reverse-engineering, where the measured dimensions can be used to construct an auger of similar dimensions and shape. We used MeshLab [2] to successfully validate the dimensions of the reconstructed 3D model (see figure 5.8).

Figure 5.4: Subset of acquired RGB images from the WR/auger dataset.

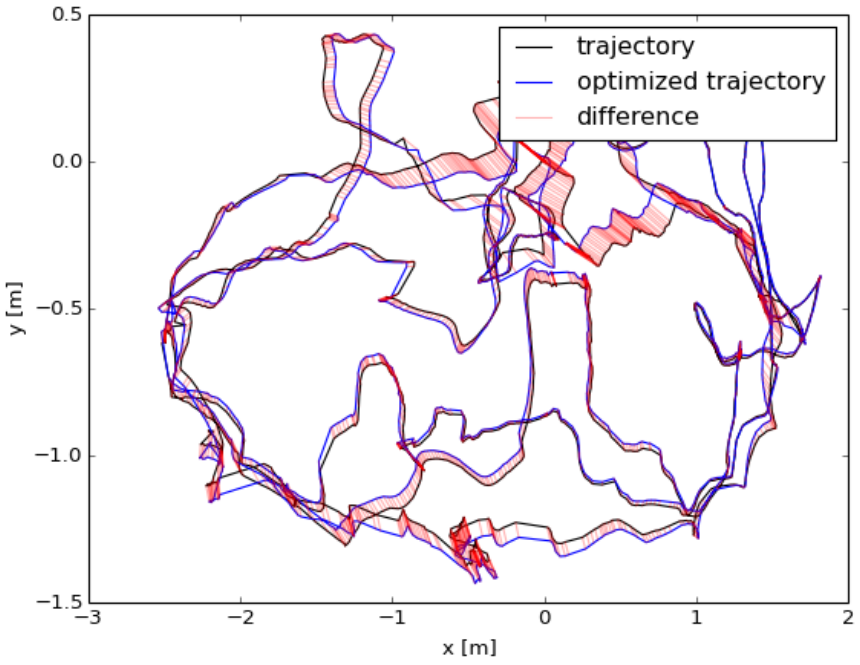


Figure 5.5: Estimated camera trajectory of the WR/auger dataset before and after submap-based bundle adjustment.

Figure 5.6: Different views of the reconstructed 3D model of the WR/auger dataset.

Figure 5.7: More views of the reconstructed 3D model of the WR/auger dataset.

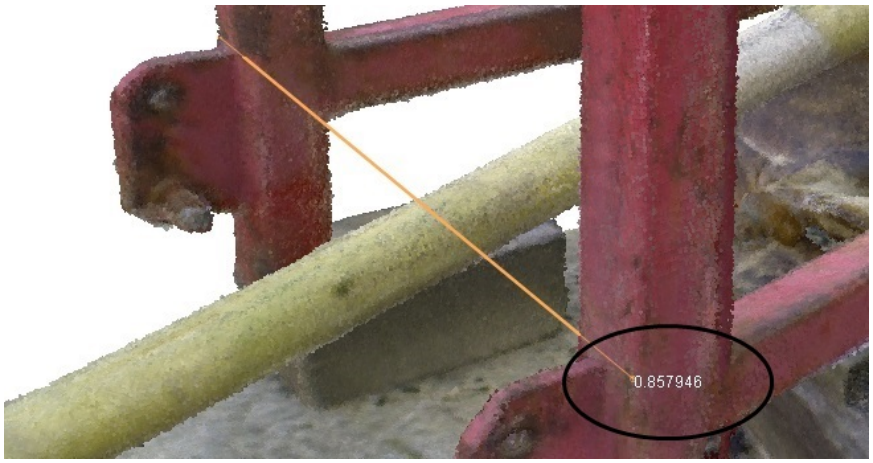


Figure 5.8: The width of the reconstructed 3D model of the WR/auger dataset measured using Meshlab [2].

5.2.2 Lawn tractor

The workpiece of interest in the WR/lawn dataset is a riding mower of size 0.94 m x 1.25 m x 2.23 m. Figure 5.9 depicts a subset of the acquired RGB images.

2167 frames of the lawn tractor were acquired in two loops around the object. While we concentrated on the lower parts in the first loop, we used the second loop to acquire frames of the upper parts with an additional close-up on the seat and the instrument panel. The optimized camera tractory with a length of 46.56 m is visualized in figure 5.10 together with the unoptimized trajectory. We performed submap-based bundle adjustment with 216 submaps in 184 s on the obtained SLAM graph, which consists of 2167 camera poses, 179616 landmarks and 1124111 observations.

The finally obtained 3D model provides an overall good visual quality and is illustrated in figures 5.11 and 5.12. However, while the global drift at a larger scale is quite limited, smaller parts like the instrument panel and the steering wheel show a poor alignment of frames.

We can imagine a potential application scenario, where the 3D model is sent to a supplier to construct customized tools. By utilizing the exact measurements of the individual mower, it possible to design tools that perfectly fit the mower's characteristics. Moreover, it is of course also possible to perform a thorough visual inspection using the reconstruction.

Figure 5.9: Subset of acquired RGB images from the WR/lawn dataset.

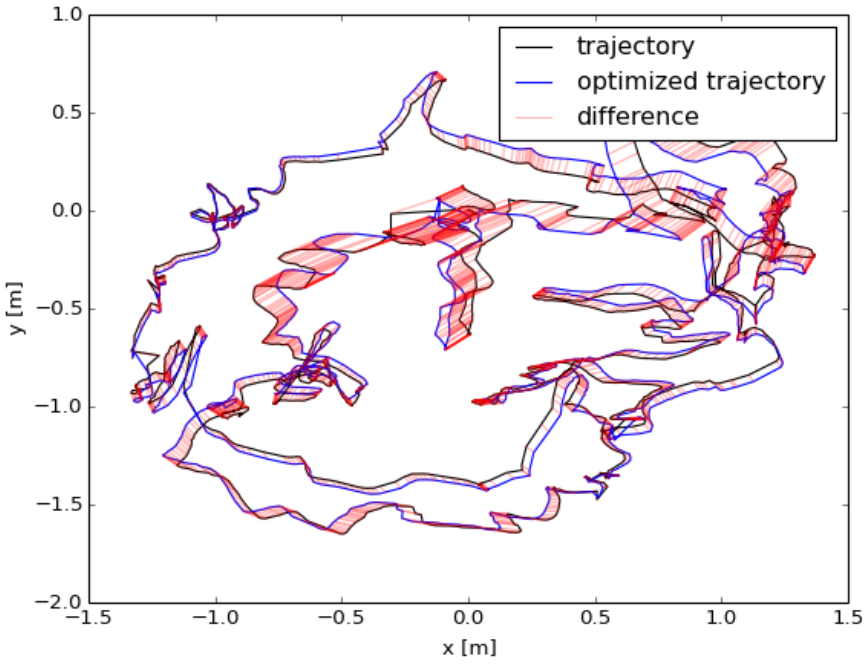


Figure 5.10: Estimated camera trajectory of the WR/lawn dataset before and after submap-based bundle adjustment.

Figure 5.11: Different views of the reconstructed 3D model of the WR/lawn dataset.

Figure 5.12: More views of the reconstructed 3D model of the WR/lawn dataset.

5.2.3 Tractor

In the WR/tractor dataset, 2087 frames of an older model of a Renault farm tractor were acquired. The object of dimensions 0.99 m x 1.30 m x 3.19 m represents the visually most complex workpiece and exhibits more small details than the WR/auger and WR/lawn

datasets, as it can be seen in figure 5.13.

The 3D reconstruction system constructed a map of 2087 camera poses, 137657 landmarks and 1063204 observations, which was bundle adjusted in 178 s using a partitioning into 208 submaps. Figure 5.14 depicts the estimated camera trajectory before bundle adjustment and the optimized trajectory of length 40.62 m.

As the camera trajectory consists basically of two simple loops around the tractor, there is no RGB-D data for some parts of the tractor. This results in a model with a sparse surface at the wheel guards right next to the seat (on both sides) and at the front part of the engine hood. Different views of the reconstructed 3D model are shown in figures 5.15 and 5.16. In figure 5.17, the model is visualized together with the corresponding estimated camera trajectory.

Similar to the previous datasets, we can use the reconstruction for visual inspection as well as for constructing customized tools for the tractor.

Figure 5.13: Subset of acquired RGB images from the WR/tractor dataset.

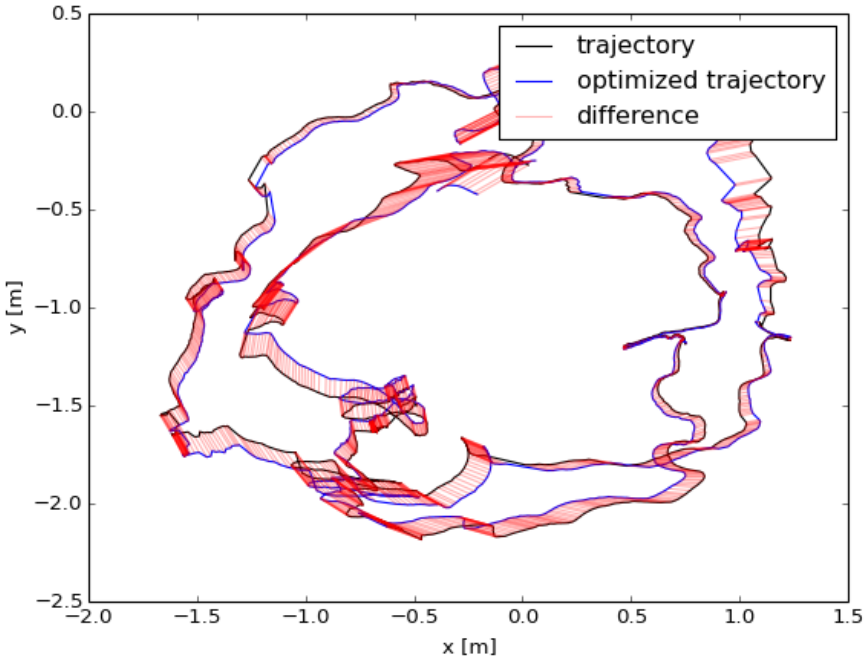


Figure 5.14: Estimated camera trajectory of the WR/tractor dataset before and after submap-based bundle adjustment.

Figure 5.15: Different views of the reconstructed 3D model of the WR/tractor dataset.

Figure 5.16: More views of the reconstructed 3D model of the WR/tractor dataset.

Figure 5.17: Reconstructed 3D model of the WR/tractor dataset together with the corresponding estimated camera trajectory.

6 Conclusion and Future Work

In this thesis, we developed an RGB-D-based 3D reconstruction framework with a special emphasis on out-of-core bundle adjustment. Our approach is suitable for 3D workpiece reconstruction as well as for the reconstruction of arbitrary scenes.

To achieve this, we first acquire RGB-D data of the scene using a hand-held Asus Xtion Pro Live RGB-D sensor. We preprocess the acquired depth map with a bilateral filter and cut off large depth values to account for increasing sensor noise with distance.

The camera pose for each frame is determined using feature-based 3D alignment. We detect and extract distinctive 2D feature points from RGB images, which allow to match images pairwise. By combining 2D feature points with depth information, we obtain 3D point correspondences of image pairs. Robust 3D alignment based on RANSAC removes outliers in these correspondences and computes the relative 3D alignment between the two frames.

A mapping component maintains a map of the environment consisting of camera poses, 3D landmarks and observations. Initial estimates for the absolute camera poses are determined by frame-to-frame tracking. Additionally, loop closures in the camera trajectory are detected by performing 3D alignment with 20 uniformly sampled previous frames.

In order to reduce the effects of accumulated drift and inaccuracies in the map over time, bundle adjustment is integrated. Instead of minimizing the 2D reprojection errors, we minimize the 3D alignment error, which is defined as the difference between measured and predicted 3D position of a 3D landmark in local camera coordinates. These additional depth constraints improve robustness, convergence behaviour and accuracy.

Because full bundle adjustment optimizes the entire map at once and consequently becomes inefficient for an increasing amount of data, we developed a new out-of-core bundle adjustment approach. We efficiently solve large bundle adjustment problems by the novel combination of a submap-based approach with the minimization of 3D alignment errors. Our submap-based approach consists of four stages. First, we partition the bundle adjustment problem into several submaps. Afterwards, we optimize the submaps independently to guarantee local accuracy. Next, we perform an efficient global alignment of the submaps. Based on the results of the global alignment, the submaps are optimized internally with fixed separators in the final stage.

Using the camera poses optimized in bundle adjustment, we finally integrate the RGB-D frames into an octree-based 3D representation to generate a dense 3D model.

We quantitatively evaluated our 3D reconstruction framework, which operates at a frame rate of 2 Hz, using the TUM RGB-D benchmark. Compared to full bundle adjustment, our submap-based bundle adjustment method improves the runtime by 32% on average and by even more than 80% for large datasets. With an average absolute trajectory error (ATE) of 0.047 m over all evaluated sequences, our method approaches the accuracy of full bun-

dle adjustment and outperforms the RGB-D SLAM system with an ATE of 0.054 m. We found empirically, that partitioning a given dataset into roughly 10 submaps per 100 contained frames is reasonable when using our approach.

Finally, we showed qualitative results of a soil auger, a lawn tractor and a small farm tractor as workpieces. With a compelling visual quality and metric accuracy, the reconstructed 3D models are suitable for visual inspection, for measuring tasks and even for reverse-engineering.

To enhance the developed framework, there are different possible directions of future research:

- With a frame rate of 2 Hz, there is room for efficiency improvements of the incorporated 3D reconstruction system.
  Except of SiftGPU for feature detection, we do not make use of GPGPU programming techniques. Using such approaches, a major speed-up of feature matching and 3D alignment estimation is possible.
  While we use RANSAC to robustly estimate the 3D alignment between 3D point correspondences, it can be replaced by the faster approach of PROSAC [11].
  Further, it is possible to incorporate special approaches for appearance-based place recognition like FABMAP [12]) into our framework in order to facilitate a more efficient detection of loop closures.
- By utilizing an octree-based model representation, we can easily generate a metrically accurate 3D point cloud of the model with an acceptable visual quality.
  However, mesh-based representations, possibly with textures from RGB images, are more suitable to obtain a visually more pleasing model and to conduct a visual inspection.
  To efficiently cope with sensor noise, a probabilistic volumetric representation like Octomap [50] is of special interest.
- The proposed submap-based bundle adjustment method is designed for offline optimization after all RGB-D input frames have been processed and aligned. To head towards online bundle adjustment, which is performed during the 3D reconstruction as a background process, it is possible to recursively partition the bundle adjustment problem into a fully hierarchical tree of submaps. Therefore, recent work of Ni and Dellaert [35, 36] is of special interest.

A Appendix

A.1 List of abbreviations

ATE Absolute Trajectory Error

ICP Iterative Closest Point

LM Levenberg-Marquardt

LS Least Squares

LLS Linear Least Squares

NLS Non-linear Least Squares

SLAM Simultaneous Localization And Mapping

RANSAC RANdom SAmple Consensus

ORB Oriented FAST and Rotated BRIEF

PROSAC PROgressive SAmple Consensus

RMSE Root Mean Squared Error

SfM Structure from Motion

SIFT Scale Invariant Feature Transform

SURF Speeded-Up Robust Features

SVD Singular Value Decomposition

TSDF Truncated Signed Distance Function

Bibliography

[1] Asus Xtion Pro Live product webpage. http://www.asus.com/Multimedia/Xtion_PRO_LIVE/. Accessed: June 11, 2013.

[2] MeshLab. http://meshlab.sourceforge.net/. Accessed: August 31, 2013.

[3] OpenNI framework. http://www.openni.org/. Accessed: May 4, 2013.

[4] Rapidform case study: Speeding the design process for NASA's SIERRA Unmanned Aerial Vehicle (UAV). http://www.rapidform.com/success-stories/aerospace-defense/. Accessed: June 28, 2013.

[5] S. Agarwal and K. Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc.

[6] K. Arun, T. Huang, and S. Blostein. Least-squares fitting of two 3-D point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700, 1987.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359, 2008.

[8] P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[9] J.-Y. Bouguet. Camera calibration toolbox for MATLAB, 2004.

[10] G. Bradski. The OpenCV library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[11] O. Chum and J. Matas. Matching with PROSAC-progressive sample consensus. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 220–226. IEEE, 2005.

[12] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[13] T. Davis. *Direct methods for sparse linear systems*. Siam, 2006.

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

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

[16] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012.

[17] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3D visual SLAM with a hand-held RGB-D camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, volume 2011, 2011.

[18] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. *Photogrammetric computer vision*, 2, 2006.

[19] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[20] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth mapping using projected patterns, oct 2008. *WO Patent WO/2008/120217*, 2008.

[21] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[22] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press, 2000.

[23] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25, 2010.

[24] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[25] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[26] K. Konolige. Large-scale map-making. In *Proceedings of the National Conference on Artificial Intelligence*, pages 457–463. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[27] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[28] M. Lourakis and A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2, 2009.

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

[30] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. springer, 2003.

[31] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[32] R. Newcombe and A. Davison. Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1498–1505. IEEE, 2010.

[33] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136. IEEE, 2011.

[34] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[35] K. Ni and F. Dellaert. Multi-level submap based SLAM using nested dissection. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2558–2565. IEEE, 2010.

[36] K. Ni and F. Dellaert. HyperSfM. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 144–151. IEEE, 2012.

[37] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3d reconstruction. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[38] K. Ni, D. Steedly, and F. Dellaert. Tectonic SAM: Exact, out-of-core, submap-based SLAM. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1678–1685. IEEE, 2007.

[39] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[40] G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Robotics Science and Systems Conference*, pages 1–8, 2009.

[41] F. Steinbrucker, J. Sturm, and D. Cremers. Real-time visual odometry from dense RGB-D images. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 719–722. IEEE, 2011.

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

[43] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

[44] J. Sturm, S. Magnenat, N. Engelhard, F. Pomerleau, F. Colas, W. Burgard, D. Cremers, and R. Siegwart. Towards a benchmark for RGB-D SLAM evaluation. In *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf.(RSS)*, pages 1–3, 2011.

[45] R. Szeliski. Rapid octree construction from image sequences. *CVGIP Image Understanding*, 58:23–23, 1993.

[46] R. Szeliski. *Computer vision: Algorithms and applications*. Springer, 2010.

[47] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. *Vision algorithms: theory and practice*, pages 153–177, 2000.

[48] T. Tykkala, C. Audras, and A. Comport. Direct iterative closest point for real-time visual odometry. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2050–2056. IEEE, 2011.

[49] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT), 2007.

[50] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010.