Monocular Camera Path Estimation Cross-linking Images in a Graph Structure

Vorgelegt von

Dipl.-Ing. Cornelius Wefelscheid
aus Essen

Von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften

– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender: Prof. Dr. rer. nat. Manfred Opper
1. Gutachter: Prof. Dr.-Ing. Olaf Hellwich
2. Gutachter: Prof. Dr.-Ing. Jan-Michael Frahm
3. Gutachter: Prof. Dr.-Ing. Helmut Mayer

Tag der wissenschaftlichen Aussprache: 10. Juni 2013

Berlin 2013
D 83

For Maria

ZUSAMMENFASSUNG

Aus den Aufnahmen mehrerer geeigneter Bilder können die 3D Informationen von einem Gegenstand oder einer Szene im Computer rekonstruiert werden. Für jedes Bild wird die Position im Raum sowie die Orientierung berechnet. Im Bereich Computer Vision wird diese Fragestellung seit mehr als zwei Jahrzehnten erfolgreich bearbeitet. Es entstanden in den letzten Jahren die ersten kommerziellen Produkte aus dem Bereich *Structure from Motion*, zu deutsch Struktur aus Bewegung. Mit vielen Anwendungsmöglichkeiten in den Gebieten des *Reverse Engineering*, der Archäologie und dem Erstellen von digitalen Stadtmodellen bieten diese Verfahren eine kostengünstige Alternative zu bestehenden Lasermesssystemen.

In dieser Arbeit wird eine Verarbeitungskette zum Erstellen von 3D Rekonstruktionen aus Bildsequenzen präsentiert, welche bestehende Verfahren robuster, zuverlässiger und schneller macht. Durch die modulare Herangehensweise, aufbauend auf einer einheitlichen relationalen Datenstruktur, können viele Neuerungen und Verbesserungen einfach und unabhängig in die Verarbeitungskette integriert werden. Jedes Modul erhält dabei Zugriff auf alle Daten. In jedem Bild werden markante Punkte erkannt und paarweise Punktkorrespondenzen zwischen zwei Bildern hergestellt. Es wurde ein neues Verfahren entwickelt, welches den nächsten Nachbarn in einem hoch dimensionalen Raum findet und auf die Beschreibung von markanten Punkten angewendet wird. Verschiedene neue Verfahren wurden in die Verarbeitungskette integriert, um effizient Schleifenschlüsse zu erkennen. Die relative Orientierung zwischen zwei Bildern wird zuverlässig mit Hilfe einer hierarchischen Clusteranalyse bestimmt. Aus Bilddrillingen, welche die relative Orientierung von Bildpaaren beinhalten, wird ein Graph erstellt, aus dem für jedes Bild die Position und Orientierung abgeleitet werden kann. Diese werden mit Hilfe von nichtlinearen Optimierungsmethoden verbessert, um ein möglichst genaues Ergebnis zu erzielen. Die Verfahren wurden auf verschiedenen Datensätzen mit einem besonderen Augenmerk auf die absolute und relative Genauigkeit evaluiert. Die Ergebnisse übertreffen Verfahren, welche als Stand der Technik gelten.

ABSTRACT

Starting from several suitable images of an object or a scene, the 3D information can be reconstructed. For each image the pose of the camera is estimated. The corresponding research field within computer vision is known as structure from motion. Intensive research for more than two decades resulted in first commercial applications covering the area of reverse engineering, archaeology and city modelling. Thus, structure from motion is going to become a low cost alternative technology to laser based systems.

In this work, a toolchain for 3D reconstruction from an image sequence was developed. It contains new algorithms to enhance current approaches concerning robustness, reliability and speed. A modular approach was chosen enabling easy and independent integration of new algorithms. The underlying relational data structure is essential to access all data at any stage of the toolchain. As a first step interest points are detected in each image to establish point correspondences between two images. A new algorithm is presented to find the nearest neighbor in a high dimensional space, applicable to the description of interest points. Different approaches to efficiently detect loop closures are integrated. The relative orientation between image pairs is computed in a robust manner following a hierarchical clustering approach. A graph containing image triplets is then constructed from a set of relative orientations and the pose of each image is estimated from the graph. Utilizing non-linear optimization techniques, the initial pose of the images is refined to establish a more accurate solution. The complete toolchain is evaluated on several datasets with a strong focus on precision and accuracy. Current state of the art methods could be outperformed.

ACKNOWLEDGEMENTS

First of all I would like to thank my supervisor Professor Hellwich for giving me the opportunity to work in his group and letting me work on such an interesting topic. He gave me the freedom to follow my own ideas and at the same time he always had an open door to discuss any questions. I would like to thank Professor Frahm for reviewing my work and coming to Berlin for the disputation. I also want to thank Professor Mayer for reviewing the thesis and the valuable feedback.

I want to thank my colleagues Tilman and Matthias with whom I shared an office. We had a lot of fruitful discussions, even some of them found their way into this work. I want to thank Marion for doing my travelling expense report and saving me some extra time which I could spend more on my thesis. I want to thank all my other colleagues. This heterogeneous group of people allowed this place to become something special. So that no working day got ever boring. I thank my colleague Adam for the after work climbing events especially those at the Schwedter Nordwand I will never forget. The climbing trips recharged my batteries even in times where the research was not going as expected.

I want to thank my mother and my sister Ulrike for the final proofreading and also my entire family for always encouraging and supporting me.

Finally, my biggest thanks go to Maria. Without you the work would not be what it is.

TABLE OF CONTENTS

ZUSAMMENFASSUNG v

ABSTRACT vii

ACKNOWLEDGEMENTS ix

LIST OF FIGURES xiii

LIST OF TABLES xvii

LIST OF ABBREVIATIONS xviii

CHAPTER

I. Introduction 1

1.1 Motivation 2
1.2 Problem Statement and Contribution 3
1.3 Outline 7

II. Theoretical Background 9

2.1 Notation 9
2.2 Camera Model 11
2.3 Features 13
2.4 Non-Linear Least Squares Optimization 17
2.4.1 Levenberg-Marquardt 18
2.4.2 Conjugate Gradient 19
2.4.3 Robust Cost Functions 21
2.5 Helmert Transformation 23

III. Data Structures 25

IV. Matching and Point Identities 33

4.1 Matching 34
4.1.1 KD-Tree 35
4.1.2 Tree Based Hashing 36

4.1.3 Evaluation 38
4.1.4 Filtering 39
4.2 Point Identities 42

V. Multi View Geometry 45

5.1 Relative Orientation 46
5.1.1 Selecting Distributed Points 47
5.1.2 Five-Point Algorithm 49
5.1.3 Hierarchical Clustering 50
5.2 Camera Triplet 63
5.2.1 Scale Selection 64
5.2.2 Tile Variance within a Triplet 65
5.3 Graph Generation 65
5.3.1 Related Approaches 65
5.3.2 Edge Weight 67
5.4 Path Estimation 68
5.4.1 Betweenness Centrality 69
5.4.2 Camera Pose 69
5.4.3 Camera Pose Optimization 71
5.4.4 Merging Reconstructions 79

VI. Loop Closure 81

6.1 Image Based Loop Closure Detection 82
6.1.1 Vocabulary Trees 82
6.1.2 Variance Descriptor Analysis 83
6.1.3 Evaluation 84
6.2 Spatial Loop Closure Detection 86
6.3 Closing the Loop 88

VII. Results 93

7.1 Faller Train Station 94
7.1.1 Loop Closure 94
7.1.2 Accuracy 98
7.1.3 Precision 101
7.2 Castle-P30 102
7.3 KPM 104
7.4 Car 106
7.5 UAV 107

VIII. Conclusion and Outlook 111

LIST OF FIGURES

Figure

1.1 Camera mounted on a UAV [94]. 2

2.1 Distorted camera model. 12

2.2 Fish-eye camera model. 13

2.3 DOG for different scales and octaves [52]. 16

2.4 Area for estimating the feature description [52]. 17

2.5 Different cost functions. 22

3.1 A project contains a unique `ID`, a `title`, `description` and an acquisition `date`. 26

3.2 The data structure for an image. 27

3.3 Structure containing the calibration of a camera. 28

3.4 Structure of a point based feature. 29

3.5 A match of two features is stored by their unique feature IDs. . 29

3.6 Structure to save the relative orientation between two images as well as the depth ratio to the 3D points. 30

3.7 The relative orientations and the scales describe an image triplet. 30

3.8 Complete relational data structure containing all information necessary to compute a path with a SfM approach. 31

4.1 Illustration of two images where sparse feature points are matched. 34

4.2 Histogram of two different dimensions with high and low variance. 36

4.3 Precision and performance of TBH in contrast to BBF. 38

4.4 Epipolar geometry of two views. 39

4.5 Camera setup illustrating the trifocal tensor with point correspondences. 41

5.1 Selectable points in blue with the radius representing the weight, already chosen points in red. 48

5.2 General hierarchical clustering scheme. 51

5.3 Top view of the simulation setup with degenerate points in red. 56

5.4 Distribution of the relative translation. 57

5.5 Comparison between CCHC and RANSAC (5-point algorithm). 59

5.6 Comparison between CCHC and RANSAC (8-point algorithm). 60

5.7 Comparison between CCHC and RANSAC (5-point algorithm) with outlier and noise=2.0. 61

5.8 Comparison between CCHC and RANSAC (8-point algorithm) with outlier and noise=2.0. 62

5.9 Camera triplet. 63

5.10 Camera triplet with scale. 64

5.11 Graph representation, with yellow nodes containing triplets and red nodes representing a single camera position [94]. 66

5.12 Graph with additional sink camera nodes in blue. 70

5.13 Graph with additional group nodes in green. 70

5.14 Inverse point parametrization with source and destination camera [95]. 73

5.15 Optimization objects interact with different measurement types. 74

5.16 Setup of points (blue) and cameras (red) for the simulated data [95]. 76

5.17 Applying normally distributed noise on the camera positions (100 cameras) [95]. 77

5.18 Changing the number of cameras with constant amount of noise on the camera position ($\sigma = 0.5$) [95]. 77

5.19 Three mammoths acquired at the American Museum of Natural History [95]. 78

5.20 Convergence of the mammoth dataset [95]. 78

6.1 Example scene which might mislead an appearance based loop closure detection. 81

6.2 Appearance similarity matrix (VDA) of KPM dataset. 84

6.3 Number of matches of the KPM dataset. 85

6.4 ROC for VDA and vocabulary trees (VOC). 86

6.5 Sparse point cloud with camera poses after initial path computation. 89

6.6 Sparse point cloud with camera poses after loop closure matched. 89

6.7 Spatial similarity matrix for loop closure. 90

6.8 Closure of large gap in loop with 10000 cameras. 91

7.1 Ground truth measurement adjustment (JAG3D [47]). 95

7.2 FTS with marked coordinates of GCP. 95

7.3 Example of a wrongly identified loop closure by VDA and VOC. 96

7.4 Loop closure analysis of the FTS dataset. 97

7.5 Reconstructed camera path with sparse point cloud. 98

7.6 RMSE of the FTS dataset for different BA configurations corresponding to Tables 7.2 and 7.3. 99

7.7 Precision of the FTS dataset. 102

7.8 Sample image of the Castle-P30 dataset. 103

7.9 Distance and angle error of the Castle-P30 dataset. 103

7.10 Sample images of the KPM dataset. 104

7.11 Top view of the court yard of KPM [25] with an approximate camera path in black. 104

7.12 Precision of the KPM dataset. 105

7.13 Acquisition setup with two GoPros on a car. 106

7.14 Driven path in blue, reconstructed path in red [25]. 107

7.15 House acquired by a UAV. 108

7.16 Camera path and quasi dense point cloud for the UAV dataset. . 109

8.1 Computational power comparison between GPUs and CPUs [68]. 112

8.2 Quadrifocal camera setup to capture deformable objects. 114

LIST OF TABLES

Table

2.1 Notation of the variables used in the thesis. 10

2.2 Weight calculation for different cost functions 23

5.1 RMSE of the relative translation difference to ground truth. . . 58

5.2 Evaluation of different edge cost functions. 68

7.1 Speed comparison of VDA and VOC in seconds. 96

7.2 RMSE in mm compared to ground truth with filtered feature matches (filter from Section 4.1.4.3). The truncation of Huber T. is applied at twice the threshold. 99

7.3 RMSE in mm compared to ground truth with all feature matches (only epipolar filter applied). The truncation of Huber T. is applied at twice the threshold. 100

7.4 MAE and RMSE of Inverse BA, BA and VisualSFM. 103

LIST OF ABBREVIATIONS

BA	Bundle Adjustment
BBF	Best Bin First
CCHC	Cross Compare Hierarchical Clustering
COO	coordinate list
CG	Conjugate Gradient
DOG	difference-of-Gaussians
EKF	extended Kalman filter
FPS	frames per second
FTS	Faller Train Station
GPS	Global Positioning System
GCP	ground control points
HC	Hierarchical Clustering
KLT	Kanade-Lucas-Tomasi
KPM	Königliche Porzellan Manufaktur
LIDAR	LIght Detection And Ranging
LM	Levenberg-Marquardt
NN	nearest neighbor
MAE	mean absolute error
MSER	maximally stable extremal regions
RANSAC	RANdom SAmple Consensus
RMSE	root mean square error
ROC	Receiver Operating Characteristic

PMVS	Patch-based Multi-view Stereo
OpenOF	Open Optimization Framework
SBA	Sparse Bundle Adjustment
SSBA	Simple Sparse Bundle Adjustment
PBA	Parallel Bundle Adjustment
SfM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SLR	single-lens reflex
SVD	Singular Value Decomposition
TBH	Tree Based Hashing
TF-IDF	Term Frequency Inverse Document Frequency
TF	Term Frequency
UAV	Unmanned Aerial Vehicle
VDA	Variance Descriptor Analysis
VD	Variance Descriptor

CHAPTER I

Introduction

In the last decades computer vision has been influencing a broad range of newly arising technologies. With the growing interest and the wide availability of digital cameras at the beginning of the 21st century, computer vision research has found its way in many consumer products. Face detection in consumer cameras or human motion sensing as a gaming control device with a Microsoft Kinect are only two of many examples. Within computer vision, a major research field focuses on the 3D reconstruction of a scene from images in a geometric sense. Measuring in images has a long history and originally belonged to the research area of photogrammetry. At former times, specifically designed cameras were used by photogrammetrists. Nowadays, high quality consumer cameras can be used for the same task leading to astonishing results. With a suitable software, expensive 3D reconstruction techniques can be made available for consumers. Even an integration into smart phones seems to be possible. A key aspect of generating a 3D model automatically is the reconstruction of the camera pose, meaning the estimation of position and orientation of each image. While the camera is moving in space, the structure of the scene can be inferred. With complete knowledge of the scene, the computation of the path is simple and vise versa. As both the path and the 3D scene are unknown in advance, they have to be estimated at the same time. In robotics, this task is known as Simultaneous Localization and Mapping (SLAM). The computer vision community refers to this problem as Structure from Motion (SfM). The key difference between SLAM and SfM are the related applications and involved constraints. SLAM is most often used in realtime applications, whereas SfM finds the most accurate solution in an offline process. Those time constraints are crucial for the choice of algorithms regarding the solution. Most SLAM approaches rely on motion models and probabilistic filters, whereas SfM approaches utilize non-linear optimization techniques. This work presents a modular approach to solve the SfM problem, with the aim to enhance reliability as well as accuracy.

Figure 1.1: Camera mounted on a UAV [94].

1.1 Motivation

Humans acquire 3D information with their two eyes. Similarly, the same information, up to a scale factor, can also be gathered by one camera which takes several images of the scene from different positions. In genereal, SfM describes the process of computing the structure of a scene captured by a moving camera. Knowing the 3D structure or the positions of the camera provides the basis for many commercial applications. In the film industry those techniques have been used in match moving software to render synthetic objects in a real world video. SfM has been also used for automatic 3D movie generation of 2D films [37]. In both examples the visual appearance of the newly created video material is important as small errors are easily identified by human eyes and reduce the enjoyment while watching the film. In contrast to the visual appearance, the geometric precision is less important in both examples. On the other hand, applications related to architecture or reverse engineering require above all a high geometric precision.

SfM approaches are a cheap alternative to expensive laser scanners. While laser scanners directly produce a 3D point cloud of the scene by measuring the distance with a laser, image based approaches need computational power to compute the 3D scene. Nevertheless, SfM has several advantages over laser scanners. Cameras are light weight and can be mounted e.g. on an Unmanned Aerial Vehicle (UAV) (see Figure 1.1). For a LIght Detection And Ranging (LIDAR) system, a much larger aircraft would be needed to create digital terrain models

due to the weight of the laser. Even though light weight laser scanners have recently been introduced and mounted on a UAV [18], the measuring range of such a light weight system is most often limited to a few meters. Recently, with the UTM-30LX from Hokuyo, a mobile laser scanner with a range of 30 m has been introduced with a weight of only 210 g, but a minor precision of ± 30 mm. As this laser scanner is designed for mobile robots, it is well suited to compute a map wherein the robot can navigate. However, it is not precise enough for measuring in point clouds. Beyond these technical disadvantages, this laser scanner is quite expensive with a price of more than 4000 €.
One major advantage of camera based systems is the usability. An inexperienced user can take images with a low cost consumer camera and can still achieve a good 3D reconstruction. Only a trained professional is able to operate a LIDAR system. In applications which aim at creating a 3D reconstruction of an object, the data acquisition with a camera is done in a few minutes whereas the operation of a laser scanner takes much longer especially if the laser has to be moved to capture the object of interest from all sides.
SfM has also been used to reconstruct much bigger areas, such as the San Marco Square in Venice or the Colosseum in Rome [2, 20]. For those reconstructions only publicly available images from foto sharing sites such as Flickr.com were used.
Another use case for SfM is the generation of digital elevation models. For larger areas, this is done by satellites or aircrafts. A similar approach can be applied for smaller areas with model airplanes or UAVs [56].
Lately, several commercial applications which solve the SfM task have been introduced: Autodesk 123D Catch (formerly Photofly), Pix4d and Photosynth are the most prominent examples. All of them create good results on many different scenes, but still face challenges on others. One can say that the SfM problem is well understood but still lacks reliability and efficiency. Further research is necessary, before the problem can be regarded as solved.

1.2 Problem Statement and Contribution

This thesis describes a solution to the following problem: Given a set of images the described toolchain computes for each image the exterior orientation or the camera as well as a sparse 3D point cloud. The focus of this work lies on image sequences, at least three images shall visually overlap. The scene which is to be reconstructed is assumed to be static, non static parts are detected as outliers. The toolchain relies on calibrated cameras, meaning that the interior orientation is known. The resulting 3D reconstruction is a metric reconstruction up to one scale factor. Restricting the images to be ordered in a sequence, allows the reconstruction of symmetric objects. Nevertheless, computing unordered images is possible but may produce errors due to wrong identified image matches. Computing the exterior orientation of the camera is referred to as computing the path of the camera. In contrast to most SLAM approaches, arbitrary motions are allowed and no motion model is assumed.

The SfM problem addresses many different subtopics which need to be solved. The topics follow the structure in which the data is processed in the pipeline. Contributions which are later described in the thesis are indicated.

- **Massive data handling**: The SfM problem cannot be solved without massive amounts of data. Handling the data efficiently simplifies the integration of new modules and the access to any data at any time in the toolchain.

  Contribution:

  > A relational data structure is proposed which enables an efficient parallel computation of the data.

- **Camera calibration**: Each camera has an individual focal length. Most often the principal point is assumed to be at the center of the image. Since consumer cameras or cameras with wide angle lens often have a radial distortion, corresponding parameters need to be computed in a calibration step.

- **Feature extraction/description**: Points which are easy to identify in different views from different perspectives are called interest points. These points shall be found and extracted in all images. A description which is invariant to geometric as well as radio metric changes within the image is computed for each feature.

- **Matching features**: Given two sets of features where each set belongs to an image, correspondences between features are established. Respecting the epipolar geometry, wrong matches are eliminated.

Contribution:

> In this subtopic a new method is developed for matching Scale Invariant Feature Transform (SIFT) features. The new method, named Tree Based Hashing (TBH), outperforms Best Bin First (BBF) regarding speed and precision.

- **Relative orientation**: With a known internal calibration, the two view geometry can be solved from five point correspondences. The solution is influenced by noise and outliers. From a set of multiple solutions the best one is chosen. Choosing the right solutions is usually done with RANdom SAmple Consensus (RANSAC).

  Contribution:

  > In this work, a hierarchical clustering approach is presented. This Cross Compare Hierarchical Clustering (CCHC) can better cope with quasi-degenerate configurations and is more robust with respect to noise.

- **Solving three view geometry**: From two relative orientations and points identified in all three images, the scale between the second and the third camera position is computed in relation to the first and second camera.

  Contribution:

  > An empirically evaluated cost function is found, which represents the reliability of a triplet. This cost is taken as a weight to generate a graph of image triplets.

- **Computing path from triplets**: A graph structure is presented which consists of image triplets. Two triplets share an edge if two of the images are identical.

  Contribution:

  > Within the graph, a metric reconstruction for each image is found by selecting an initial triplet which is supposed to be most central. From this starting triplet a shortest path to each image is selected according to the edge weight.

- **Detecting and closing loops**: Returning to a place which has been seen before is common when capturing an object. Most often a circle is closed, but more complex loops are possible. Those shall be identified and the accumulated drift is distributed.

  Contribution:

  > The loop closure detection is handled in two ways. First, an appearance based approach is presented which defines a new descriptor for each image. This new descriptor is named Variance Descriptor Analysis (VDA). Second, a spatial voting scheme finds a loop closure candidate in case an initial camera path and a sparse 3D point cloud is given.

- **Bundle Adjustment**: Bundle Adjustment (BA) optimizes the computed results with a non-linear least squares optimization procedure.

  Contribution:

  > Within this work, a new BA parametrization is introduced which reduces the number of parameters. A 3D point is not any longer free in space, but it is attached to the image where it first appeared. The only unknown parameter which needs to be estimated is the inverse depth. A non-linear least squares optimization framework, named Open Optimization Framework (OpenOF) is presented to handle large optimizations efficiently.

The toolchain should be as generic as possible meaning that the reconstruction is not limited to a special scene type. The following scenes are used for demonstration purposes in this work:

- Model house to analyse the accuracy.
- House with images acquired by a UAV.
- Skeletons at the American Museum of Natural History.
- Street scene, captured with cameras mounted on a car.
- Courtyard acquired with a camera mounted on a Segway.
- Publicly available ground truth dataset.

Parts of the thesis have been published:

> Cornelius Wefelscheid, Ronny Hänsch, and Olaf Hellwich. Three-dimensional building reconstruction using images obtained by unmanned aerial vehicles. In *Proceedings of the International Conference on Unmanned Aerial Vehicle in Geomatics (UAV-g)*, Zurich, Suisse, Sep 2011
>
> Cornelius Wefelscheid and Olaf Hellwich. OpenOF: Framework for sparse non-linear least squares optimization on a GPU. In *VISAPP*. SciTePress, 2013

Furthermore, the open source library, OpenOF, which has been developed within this work, has been used for geocoding SAR and optical satellite images:

> Olaf Hellwich, Cornelius Wefelscheid, Jakub Lukaszewicz, Ronny Hänsch, Adnan M. Siddique, and Adam Stanski. Integrated matching and geocoding of SAR and optical satellite images. In *IBPRIA 2013*. Springer, June 2013

1.3 Outline

The structure of the thesis mainly follows the presented pipeline. Chapter II covers the theoretical background. The notation which is used within this work is stated in Section 2.1. Section 2.2 describes two camera models (pinhole and spherical camera model), followed by a detailed description of the incorporated feature detector and descriptor (Section 2.3). The mathematical basis of the non-linear optimization framework, OpenOF, is given in Section 2.4. The chapter concludes with the description of a Helmert transformation. Chapter III presents the underlying relational data structure. The chosen database model supports the development of a modular approach. Feature matching between two images, as well as detecting outliers is presented in Chapter IV. From all computed matches, a reliable camera path is estimated in Chapter V where a graph of image triplets is generated to find the best initial estimate which is then optimized with a non-linear least squares method. Chapter VI investigates loop closures, which can either be detected by the visual appearance of the images or by the path which has been reconstructed by the methods presented in Chapter V. The thesis concludes with the results in Chapter VII and the conclusion and an outlook to future work in Chapter VIII.

CHAPTER II

Theoretical Background

The work is based on theoretical background described in this chapter. After a brief introduction of the mathematical notation, which is used throughout the work, two different camera models are described as those are used for different datasets. Within Section 2.3, the visual feature as well as the corresponding feature description is presented in detail. The new matching scheme of TBH as presented in Section 4.1.2 goes hand in hand with the description of a feature. Many stages of the toolchain incorporate non-linear least squares optimizations. Most often the Levenberg-Marquardt (LM) algorithm in conjunction with a Conjugate Gradient (CG) approach is used to handle large scale optimizations with thousands of parameters. Different robust cost functions are stated in Section 2.4.3 which are used in order to suppress outliers.

2.1 Notation

Throughout this work the following notation is used, if not stated otherwise: Bold letters are used for matrices and vectors. Commonly, capital letters are used for matrices, but not exclusively. Bold capital letters are used for 3D points as well, most often $\mathbf{X}$ in homogeneous coordinates. A small bold letter is used as projection into the image $\mathbf{x}$ as well in homogeneous coordinates. A 3D point in non homogeneous representation is denoted by $\hat{\mathbf{X}}$. The same holds for 2D points. Small non bold variables are used for scalars such as s or u_0. Capital non-bold variables are used as single coordinate in 3D space such as X, Y, Z. A quaternion is denoted by $\mathbf{q}$. A certain element of a vector, a matrix or a quaternion is selected by square brackets $\mathbf{q}[0]$ for the first element as common in programming languages or by a lower index, respecting the mathematical formulation $\mathbf{q}_1$. In case a lower index is already used, to denote different variables, the squared bracket notation is used. The first element of a matrix is denoted as $\mathbf{A}_{11}$. A subpart of a vector or a quaternion is selected, e.g. $\mathbf{q}[1:4]$ containing the element second to fourth,

Name	Symbols	Description
Scalar	s,f_y,f_x	scalar variable, small non-bold
Vector	$\mathbf{n}$	vector variable, small bold
Vector element	$\mathbf{n}_1$	1st element, mathematical notation
Vector element	$\mathbf{n}[0]$	1st element, programming notation
Subvector	$\mathbf{n}[1:4]$	2nd to 4th element, numpy notation
Matrix	$\mathbf{A}$	matrix variable, capital bold
Serialized Matrix	$\tilde{\mathbf{A}}$	capital bold with tilde
Matrix element	$\mathbf{A}_{11}$	first row, first column
Matrix element	$\mathbf{A}[0,0]$	first row, first column
Submatrix	$\mathbf{A}[1:4,1:4]$	see description subvector
2D point	$\hat{\mathbf{x}}$	small bold with hat
Homogeneous 2D point	$\mathbf{x}$	small bold
3D point	$\hat{\mathbf{X}}$	capital bold with hat
Homogeneous 3D point	$\mathbf{X}$	capital bold
3D coordinate	X, Y, Z	capital non-bold
Quaternion 3D point	$\tilde{\mathbf{X}}$	capital bold with tilde
Quaternion	$\mathbf{q}$	small bold
Place holder	C	camera parameters

Table 2.1: Notation of the variables used in the thesis.

considering that counting is started at zero as it is common in numpy [69]. The quaternion representation of a 3D coordinate is denoted by a tilde, for compact writing of a spatial rotation using quaternions e.g. $\mathbf{q}\tilde{\mathbf{X}}\mathbf{q}^{-1}$. In some formulas a different representation of a 3D point is needed. In this case $\mathbf{X}$, $\hat{\mathbf{X}}$ and $\tilde{\mathbf{X}}$ represent the same point in corresponding form.

$$\tilde{\mathbf{X}} = \begin{bmatrix} 0 \\ X \\ Y \\ Z \end{bmatrix} \tag{2.1}$$

In case a tilde is above a matrix, the matrix is serialized to a vector.

$$\tilde{\mathbf{A}} = [\mathbf{A}_{11}, \mathbf{A}_{12}, \mathbf{A}_{13}, \mathbf{A}_{21}, \mathbf{A}_{22}, \mathbf{A}_{23}, \mathbf{A}_{31}, \mathbf{A}_{32}, \mathbf{A}_{33}]^T \tag{2.2}$$

The notation, which is used throughout the work is summarized in Table 2.1.

2.2 Camera Model

Throughout this work, a pin hole camera model is assumed. In conjunction with BA (Section 5.4.3.1), additional distortion parameters are used, everywhere else undistorted images are assumed. A camera model describes the projection of a 3D point to a 2D image coordinate (see Figure 2.1). The 3D point $\mathbf{X}$ in homogeneous coordinates is projected by a function $f(\mathbf{X}, C)$ to a 2D image point $\mathbf{x}$. C combines all internal and external camera parameters such as $[X, Y, Z]^T$ being the camera center and $\mathbf{q}$ the quaternion representing the rotation of the camera.

$$\mathbf{x} = f(\mathbf{X}, C) \tag{2.3}$$

The internal camera parameters are denoted by f_x, f_y being each the focal length in pixels. In case of square pixels, f_x and f_y are equal. The skew parameter is represented by s. $[u_0, v_0]^T$ is the principal point. All those parameters are found in the calibration matrix $\mathbf{K}$.

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

Commonly, the projection is described by matrix multiplication (Equation 2.5), with $\hat{\mathbf{C}} = [X, Y, Z]^T$ combining the parameters of the camera center and $\mathbf{R}$ being the rotation from world to camera coordinate system and $\mathbf{I}$ is the 3×3 identity matrix. $\mathbf{K}$, $\mathbf{R}$ and $\hat{\mathbf{C}}$ can be combined to the projection matrix $\mathbf{P}$.

$$\mathbf{x} = \mathbf{P}\mathbf{X}_w = \mathbf{K}\mathbf{R}[\mathbf{I}|\text{-}\hat{\mathbf{C}}]\mathbf{X}_w \tag{2.5}$$

The projection can also be described as a quaternion multiplication, transforming a point $\mathbf{X}_w$ from world coordinate system into a 3D point in camera coordinate system $\mathbf{X}_c$.

$$\tilde{\mathbf{X}}_c = \mathbf{q}(\tilde{\mathbf{X}}_w - \tilde{\mathbf{C}})\mathbf{q}^{-1} \tag{2.6}$$

The calibration matrix projects the point into the image plane resulting in sensor coordinates.

$$\mathbf{x} = \mathbf{K}\hat{\mathbf{X}}_c \tag{2.7}$$

In case of a radially distorted camera, Equation 2.7 is extended by k_1, k_2, k_3 being the coefficients of a polynomial equation. Assuming $\mathbf{x}_d = \mathbf{x}$ is the distorted image

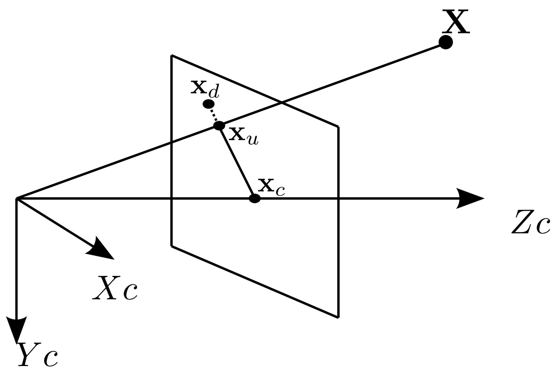


Figure 2.1: Distorted camera model.

point from Equation 2.7, the distance to the distortion center is computed as

$$r = \sqrt{(\hat{\mathbf{x}}_d - \hat{\mathbf{x}}_c)^T(\hat{\mathbf{x}}_d - \hat{\mathbf{x}}_c)}\ . \tag{2.8}$$

Usually, it is assumed that the distortion center is the principal point:

$$\mathbf{x}_c = [u_0, v_0, 1]^T \tag{2.9}$$

The undistorted image point $\mathbf{x}_u$ can be computed as

$$\mathbf{x}_u = \mathbf{x}_d + (\mathbf{x}_d - \mathbf{x}_c)(k_1 r^2 + k_2 r^4 + k_3 r^6)\ . \tag{2.10}$$

If necessary, the radial distortion model can be combined with a tangential distortion model [28]. An undistorted image is rendered by formulating the distortion model in respect of the undistorted image point. The color information of each pixel of the new image is taken from the distorted image. The new color value of the undistorted image is computed by transferring the point with the distortion model to the original image. As the point has a subpixel accuracy, the color information is interpolated between the surrounding pixels.
Recently, wide-angle fisheye lenses have gathered much attention as they are part of low cost sports cameras such as the GoPro HD HERO 2 camera. Their spherical lens produces a strong distortion, which cannot be modeled with Equation 2.10. Such a fish-eye camera needs a different distortion model (see

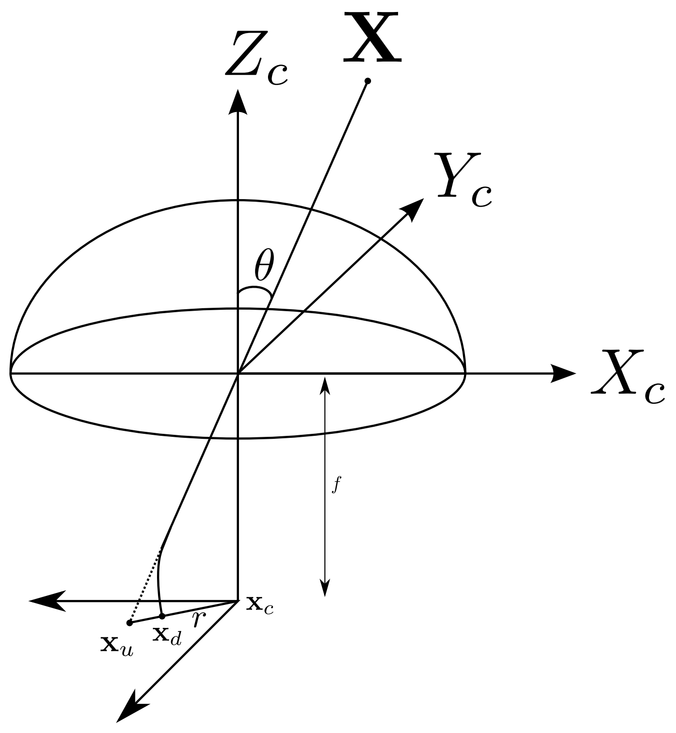


Figure 2.2: Fish-eye camera model.

Figure 2.2) [35]. The corresponding formula is given in the following equation.

$$\mathbf{x}_d = \frac{1}{r}(\theta + k_1\theta^3 + k_2\theta^5) \begin{bmatrix} f_x \\ f_y \\ 1 \end{bmatrix} (\mathbf{x}_u - \mathbf{x}_c) + \mathbf{x}_c \tag{2.11}$$

In this case the radius r and the angle θ are defined as

$$r = \sqrt{(\mathbf{x}_u - \mathbf{x}_c)^T(\mathbf{x}_u - \mathbf{x}_c)} \tag{2.12}$$

and

$$\theta = \arctan\left(\frac{2r}{f_x + f_y}\right) . \tag{2.13}$$

The parameters k_1, k_2 for the spherical distortion model and the parameters k_1, k_2, k_3 for the radial distortion model are estimated in a calibration procedure.

2.3 Features

Identifying features in images is an essential task in many computer vision applications. Features, as a review of the image processing literature shows, can be divided in global and local features. Local features can be further divided in point, shape or region based features [6, 19, 39, 52, 55, 58, 59, 60, 75]. This work focuses on point based features, also known as distinct interest points, which have been proven in the past to be most successful for camera path estimation. Most of the geometric relations used in SfM are based on the projection of a 3D point into an image. Region based approaches, like maximally stable extremal regions (MSER), suffer from the lack of a stable center point. Perspective distortions move the center point when changing the viewing angle, thus knowledge of the orientation of the region has to be incorporated to compute a stable center point. Knowing that a region is a rectangle can solve the orientation problem [96]. In case of a rectangle, the center point is estimated by intersecting the two diagonals. Feature extraction can be subdivided into two parts, the detection and the description. In [57] a point based feature is defined to be invariant, stable, interpretable, distinct and seldom. The detection has to be precise in its position (stability) and a high recognition rate is essential (invariance). Furthermore, a feature needs a proper description which allows to reidentify the same feature in different images (distinctness and seldomness). The property that a feature should be interpretable is of minor importance for a SfM toolchain.

This work incorporates point based features such as SIFT, as they have proven to fulfill the task of SfM. Other features (GLOH [58], PCA-SIFT [36], SURF [6]) have been described, but none of them is superior compared to SIFT. Another feature operator with a high popularity is SURF, which is often used for realtime applications. It outperforms SIFT in respect of speed but at the expense of reliability [33]. SURF was early integrated into the well known computer vision library, OpenCV, and therefore spread in the community.

Detector and Descriptor

A good feature for a 3D reconstruction task needs to have several properties. It has to be highly distinctive to match one feature against a large set of other features correctly. On the other hand, it has to be invariant to changes of the following properties:

- **Scale**: As the user moves around the object of interest, the distance between the camera position and the object varies, so it is required to match images taken at different scales. Also close up images shall be matched with images taken from far away.

- **Translation/Rotation**: As features appear in different positions in an image, the translational change should have no impact on either the detector or the descriptor. Images which are rotated around the viewing ray should still have the same description.

- **Illumination**: A controllable illumination of an object, as it is possible under laboratory conditions, is unlikely in real world applications. When taking images e.g. with a UAV the illumination changes from one side of the object to the other side or specular highlights illuminated by the sun may occur. Therefore, the feature has to be invariant to such circumstances to a certain degree.

- **Viewpoint**: When capturing an object from different perspectives, the appearance of an interest point changes, e.g a corner can look completely different from the other side. Invariance for large viewpoint changes can be achieved for planar scenes, but only a small angle between two viewpoints can be handled considering images of vegetation. Different studies have addressed the challenge of handling view point changes [38, 62, 99]. It is assumed to be the hardest invariance.

SIFT is invariant to a certain range of variations. It has been well studied in literature and proven to be reliable. It outperforms other feature extractors and descriptors [58].

SIFT

SIFT was first described by Lowe in [51] and later in more detail in [52]. The method itself can be separated into two parts, the key point detection and the key point description.

The keypoint detection is based on finding local maxima and minima in scale-space. The scale-space represents the image structure at different levels of detail. This can be achieved by blurring the image with a Gaussian kernel. The definition of the scale-space is given in Equation 2.14 as the convolution of the kernel with the image [45].

$$L(x, y, \sigma) = G(x, y, \sigma) \star I(x, y) \tag{2.14}$$

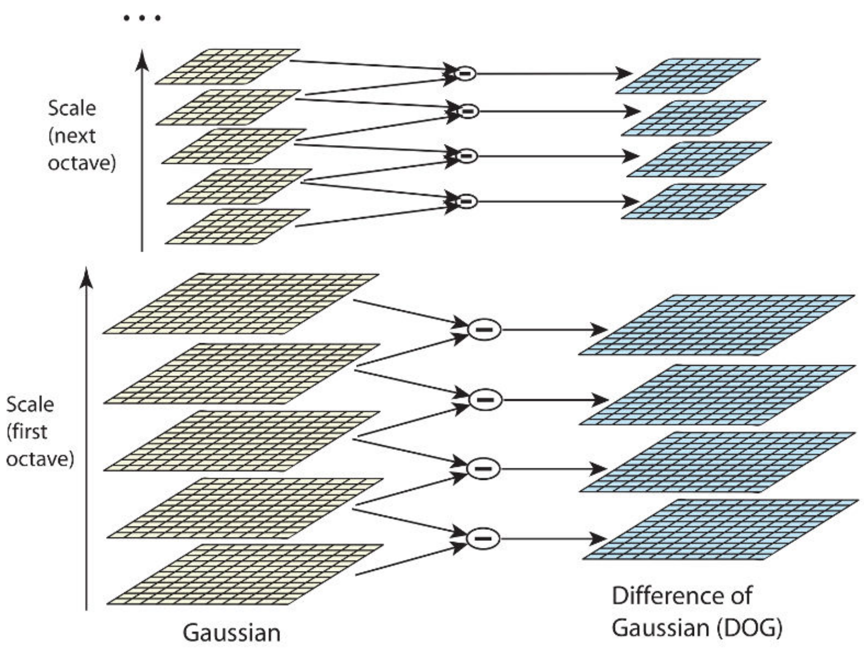


Figure 2.3: DOG for different scales and octaves [52].

Further, the Gaussian kernel is defined by

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \ . \tag{2.15}$$

For efficiency, the extrema are not detected in a linear scale-space but rather with the difference-of-Gaussians (DOG) from nearby scales using a constant factor k.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \star I(x, y) \tag{2.16}$$

Equation 2.16 can be simplified to the substraction of two scale-space levels.

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.17}$$

The scale-space can be further split up into different octaves to compute a bigger field of scales even more efficiently (see Figure 2.3). A maximum or minimum is found if all neighboring values for a voxel in the scale space grid are lower or higher, respectively. The cornerness and the contrast of each keypoint is computed to reject features with low contrast or keypoints which lie on an edge (see [52] for details).
The invariance w.r.t. scale and orientation is achieved by using the scale-space level in which the keypoint is detected as the scale. The dominant gradient

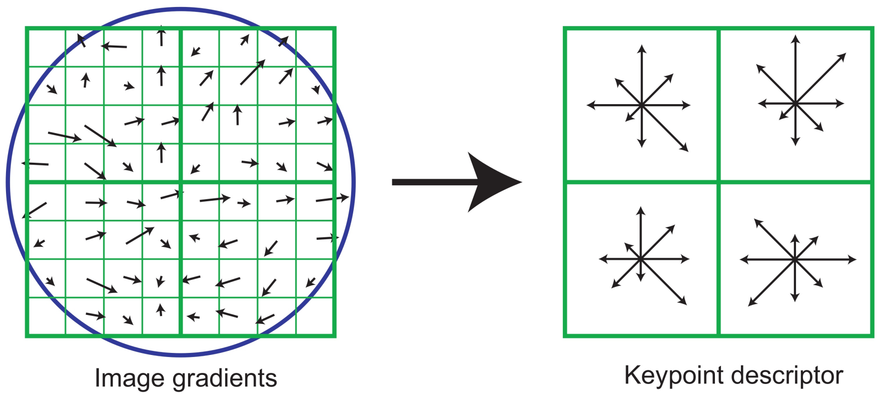


Figure 2.4: Area for estimating the feature description [52].

within a circular region around the keypoint with a radius of 1.5σ is computed, in order to achieve rotation invariance. Therefore, a gradient histogram of 36 bins containing the gradient direction is generated. The bins are filled with the magnitudes of the corresponding direction, computed from the blurred image at the underlying scale (see Equations 2.18 and 2.19). The bin with the highest score represents the orientation of the keypoint. In case of a strong second orientation reaching a score of at least 80 % of the highest score, a second keypoint with the corresponding orientation is created.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{2.18}$$

$$\theta(x,y) = \tan^{-1} \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)} \tag{2.19}$$

The description of a keypoint additionally relies on generating histograms of gradients. To describe each keypoint, the same circular area around the keypoint is examined as it was used previously. The area is partitioned in a 4×4 grid. Figure 2.4 shows an illustration of a 2×2 grid. For each tile a gradient histogram with 8 bins is computed, leading to a descriptor size of $4 \times 4 \times 8 = 128$. Rotation invariance is achieved by rotating the coordinates of the area used to compute the descriptor by the main orientation of the keypoint. Illumination changes are handled by normalizing the descriptor [52].

2.4 Non-Linear Least Squares Optimization

Within a SfM approach many different algorithms are computed with a direct linear transformation, e.g. the 8-point algorithm for computing the fundamental matrix. For the latter, the solution minimizes the algebraic error. Usually, the user is interested in an optimal solution regarding a geometric cost function. Non-linear optimization techniques are necessary to minimize the geometric error. In the last years, the LM algorithm has been most commonly used especially in combination with BA. BA is one of the main algorithms in most SfM approaches, and is used as a final optimization step. In Section 5.4.3.3 a general framework for non-linear least squares optimization is presented which is based on theory described in Sections 2.4.1 and 2.4.2 which has been published in [95].

2.4.1 Levenberg-Marquardt

LM is regarded as standard for solving non-linear least squares optimization. It is an iterative approach which was first developed by Levenberg in 1944 and rediscovered by Marquardt in 1963. The algorithm determines the local minimum of the sum of least squares of a multivariant function. An overview of the family of non-linear least squares optimization techniques can be found in [54]. LM combines a Gauss-Newton method with a steepest descent approach. Given a parameter vector $\mathbf{x}$, the task is to find a vector $\mathbf{x}_{min}$ that minimizes the function $f(\mathbf{x})$ which is defined as the sum of squares of the vector function $r(\mathbf{x})$.

$$\mathbf{x}_{min} = \arg\min_{\mathbf{x}} f(\mathbf{x}) = \arg\min_{\mathbf{x}} \frac{1}{2}\sum_{i=0}^{m} r_i(\mathbf{x})^2 \tag{2.20}$$

As the function is usually not convex, a suitable initial solution $\mathbf{x}_0$ needs to be provided. Otherwise, LM will not converge to the global optimum, but gets stuck in a local minimum, which might be far away from the optimal solution. In each iteration i, $f(\mathbf{x})$ is approximated by a second order Tailor series expansion.

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}_i) + f(\mathbf{x}_i)'(\mathbf{x} - \mathbf{x}_i) + \frac{1}{2}f(\mathbf{x}_i)''(\mathbf{x} - \mathbf{x}_i)^2 \tag{2.21}$$

To find the minimum, the first derivative of Equation 2.21 is set to zero, resulting in

$$\frac{\partial \tilde{f}(\mathbf{x})}{\partial \mathbf{x}} = f(\mathbf{x}_i)' + f(\mathbf{x}_i)''(\mathbf{x} - \mathbf{x}_i) = \mathbf{0}\ . \tag{2.22}$$

The first order derivative at the point $\mathbf{x}_i$ can be replaced by the transposed Jacobian $\mathbf{J}^T$ multiplied with the residual. The second order derivative is approximated by the Hessian matrix given as $\mathbf{J}^T\mathbf{J}$. The parameter update

$\mathbf{h} = \mathbf{x} - \mathbf{x}_i$ in each iteration is conducted by solving the normal equation.

$$\mathbf{J}^T\mathbf{J}\mathbf{h} = -\mathbf{J}^T r(\mathbf{x}_i) \tag{2.23}$$

To include the gradient descent approach, Equation 2.23 is extended by a damping factor μ.

$$(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\mathbf{h} = -\mathbf{J}^T r(\mathbf{x}_i) \tag{2.24}$$

For $\mu \to 0$ LM applies the Gauss-Newton method and for $\mu \to \infty$ a gradient descent step is executed. If the update step does not result in a smaller error, the damping factor is increased pushing the parameters further towards the steepest descent. Else, the update is performed and the damping factor is reduced. In cases with a known covariance of the measurements, Equation 2.24 can be extended to

$$(\mathbf{J}^T\mathbf{\Sigma}^{-1}\mathbf{J} + \mu\mathbf{I})\mathbf{h} = -\mathbf{J}^T\mathbf{\Sigma}^{-1} r(\mathbf{x}_i)\ . \tag{2.25}$$

Instead of the Euclidean norm the Mahalanobis distance is minimized. Solving the normal equation is often the most demanding part of the algorithm. The computation of the Jacobian matrix can be also time consuming, but fortunately each element can be computed in parallel. If the normal matrix has a defined structure,

$$\mathbf{N} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \tag{2.26}$$

with $\mathbf{D}$ being invertible, approaches such as the Schur complement can be applied [91]. In general the normal equation can be solved with a Cholesky decomposition. Another possibility is an iterative approach such as CG, which will be described in the next section.

2.4.2 Conjugate Gradient

The CG method is a numerical algorithm for solving a linear equation system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{A}$ being symmetric positive definite and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. As CG is an iterative algorithm, there exists a residual vector $\mathbf{r}_k$ in each step k defined as:

$$\mathbf{r}_k = \mathbf{A}\mathbf{x}_k - \mathbf{b}\ . \tag{2.27}$$

CG belongs to the Krylov subspace methods [67]. Krylov subspace is described by

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) = span\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\} \tag{2.28}$$

and is taken as the basis for many iterative algorithms. It has been proven that those methods terminate in at most n steps [67]. CG is a highly economical method which does not require much memory or time demanding matrix-matrix multiplication. While applying CG, a set of conjugate vectors $(\mathbf{p}_0, \mathbf{p}_1, .., \mathbf{p}_{k-1}, \mathbf{p}_k)$ is computed belonging to the Krylov subspace. In each step the algorithm computes a vector $\mathbf{p}_k$ being conjugate to all previous vectors. Only the previous vector $\mathbf{p}_{k-1}$ is needed and every other vector can be omitted, thereby saving memory. The algorithm is based on the principle that $\mathbf{p}_k$ minimizes the residual over one spanning direction in every iteration. CG is initialized with $\mathbf{r}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b}, \mathbf{p}_0 = -\mathbf{r}_0, k = 0$. In each iteration a step length α is computed, which minimizes the residual over the current spanning vector $\mathbf{p}_k$.

$$\alpha_k = -\frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k} \tag{2.29}$$

The update of the current approximate solution is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{2.30}$$

which results in a new residual

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A} \mathbf{p}_k \ . \tag{2.31}$$

The new conjugate direction is computed from Equations 2.32 and 2.33.

$$\beta_{k+1} = -\frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} \tag{2.32}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k \tag{2.33}$$

Finally k is increased. This procedure is repeated as long as $||\mathbf{r}_k|| > \epsilon$ and $k < k_{max}$. Theoretically n iterations could be necessary to solve the linear equation system, but in most cases, only a few iterations are needed to find a good approximation. Prove of convergence and a full derivation are given by [67].

For practical considerations the explicit computation of the matrix $\mathbf{A} = \mathbf{J}^T\mathbf{J}$ can be omitted. Concatenating matrix-vector multiplications for $\mathbf{J}^T$ and $\mathbf{J}$ is faster and the numerical precision is higher. Such an approach is known as least squares CG.

2.4.3 Robust Cost Functions

When applying non-linear least squares optimization, the input data has to be free of outliers. As not all outliers can be excluded, introducing robust cost functions can avoid gross failures caused by outliers. Instead of optimizing $r_i(\mathbf{x})^2$ (see Equation 2.20), $g(r_i(\mathbf{x}))$ is optimized. In the standard least squares approach, which is based on a Gaussian noise assumption, the cost function is

$$g(y) = y^2 \tag{2.34}$$

with $y = r_i(\mathbf{x})$ being the result of the function of interest. As the Gaussian noise assumption does not hold for outliers, many different cost functions are proposed in literature [28]. The most common cost functions will be evaluated in Section 5.4.3.1 regarding BA. The simplest robust cost function is the truncated L2 function which is quadratic within an inlier range and constant otherwise.

$$g(y) = \left\{ \begin{array}{ll} y^2 & : |y| < \alpha \\ \alpha^2 & : |y| \geq \alpha \end{array} \right\} \tag{2.35}$$

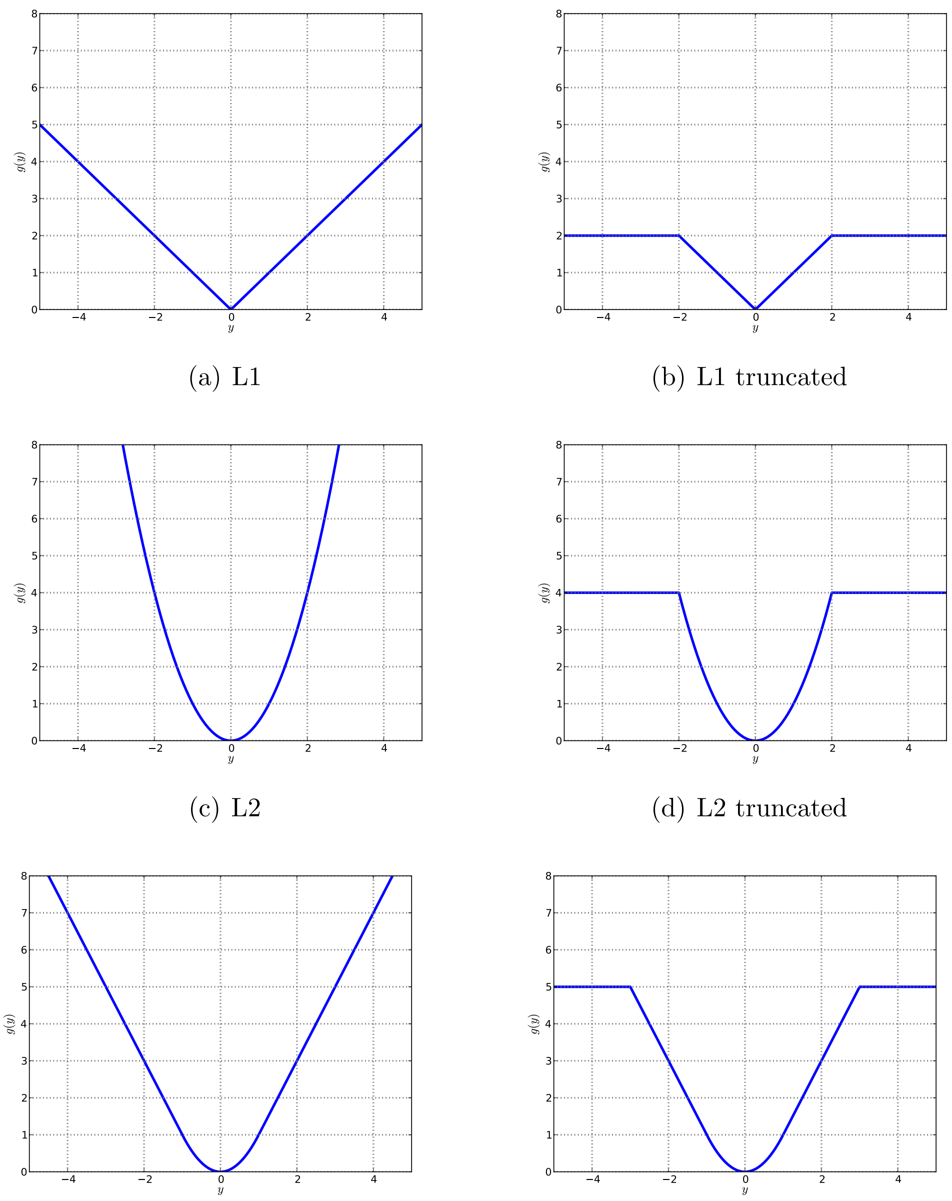
Outliers influence the optimization only with a constant cost, but it is not differentiable at $|y| = \alpha$. Another popular cost function is the L1-norm.

$$g(y) = |y| \tag{2.36}$$

This cost function is differentiable except for the origin. A combination of a quadratic and a linear cost has been introduced and named by Huber [30].

$$g(y) = \left\{ \begin{array}{ll} y^2 & : |y| < \alpha \\ 2\alpha|y| - \alpha^2 & : |y| \geq \alpha \end{array} \right\} \tag{2.37}$$

The cost is quadratic within an inlier range and linear otherwise.
Integrating a robust cost function in the LM algorithm is achieved by computing an individual weight which scales the squared cost function to the desired function. This enables the integration of arbitrary cost functions in the presented



(a) L1 (b) L1 truncated

(c) L2 (d) L2 truncated

(e) Huber (f) Huber truncated

Figure 2.5: Different cost functions.

name	$0 < \|y\| \le \alpha$	$\alpha < \|y\| \le \beta$	$\beta < \|y\|$
L1	$1/\sqrt{\|y\|}$		
L1 truncated	$1/\sqrt{\|y\|}$	$\sqrt{\alpha}/\|y\|$	
L2	1		
L2 truncated	1	$\alpha/\|y\|$	
Huber	1	$\sqrt{2\alpha\|y\| - \alpha^2}/\|y\|$	
Huber truncated	1	$\sqrt{2\alpha\|y\| - \alpha^2}/\|y\|$	$\sqrt{\alpha^2 + 2\alpha(\beta - \alpha)}/\|y\|$

Table 2.2: Weight calculation for different cost functions

LM algorithm.

$$w = \sqrt{g(y)}/|y| \tag{2.38}$$

Near the optimum for the L1 and L1 truncated case, special attention regarding divisions by zero is needed for the implementation. In Table 2.2 the explicit computation of the weight is shown for different cost functions with respect to their threshold parameters, α and β. It is assumed that $0 < \alpha < \beta$ holds. In case a model has no parameters α and β are set to infinity. The same holds for β if only α is needed. A visual reprensentation of different cost functions is given in Figure 2.5.

2.5 Helmert Transformation

When acquiring a 3D reconstruction with a monocular camera, the resulting reconstruction is unique up to seven parameters. These parameters contain a translation, rotation, and a scale. The corresponding transformation is known as Helmert transformation or 3D similarity transformation. As the reconstruction is metric, two reconstructions of the same object only differ by one Helmert transformation. Given a set of n points $\{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_{n-1}, \mathbf{X}_n\}$ and their correspondences $\{\mathbf{X}'_1, \mathbf{X}'_2, \dots, \mathbf{X}'_{n-1}, \mathbf{X}'_n\}$, there is a transformation

$$\tilde{\mathbf{X}}'_i \approx f(\tilde{\mathbf{X}}_i) = \tilde{\mathbf{t}} + s\mathbf{q}\tilde{\mathbf{X}}_i\mathbf{q}^{-1} \tag{2.39}$$

with $\mathbf{t}$ being a translation, $\mathbf{q}$ a quaternion representing a rotation and s is a scaling factor. The solution for $\mathbf{t}, \mathbf{q}, s$ can be found by solving the following least squares optimization problem.

$$\arg\min_{\mathbf{t},\mathbf{q},s} \frac{1}{2} \sum_{i=0}^{m} (\tilde{\mathbf{X}}'_i - f(\tilde{\mathbf{X}}_i))^2 \tag{2.40}$$

This is achieved using the LM algorithm described in Section 2.4.1. LM usually needs a good initial solution, otherwise the initialization can cause the optimization to get stuck in a local minimum. This is the case especially when the two point clouds are rotated by 180 degree against each other. Since the optimization is unconstrained, the scale can become negative. In case of a negative scale, the transformation causes the transformed point cloud to be mirrored at the origin. Such a situation can be avoided by setting an approximate initial rotation which is then refined in the optimization.

CHAPTER III

Data Structures

The generation of a 3D model requires handling massive amounts of data. Former approaches stored temporary results in several files [81]. This complicates the integration of new ideas. An easy and fast access to all data is needed at any point in the toolchain. As other research areas face the same challenge, extensive research has been done by database specialists. Sophisticated approaches are described in the literature to handle massive amounts of data [14]. Especially, a deliberate design of the data structure is important to handle efficient reading and writing of the data. Most popular are relational databases such as MySQL, MSSQL or SQLite. Lately, also NoSQL (not only SQL) databases have gathered much attention due to their use by prominent companies such as Google and Facebook. In contrast to relational database models, NoSQL databases have no fixed scheme and are optimized to run on clusters to handle queries from thousands of users at the same time. They are not designed to combine entries from multiple tables in one query by using joins.

In this work a relational data model, which is capable of handling all information and gives easy access to prior computed results, is used. Thus, the 3D reconstruction can be modularized without keeping every information in memory or the need to handle massive reads and writes on the harddrive. The storage and compression of data is managed by the database server. All prior computed results are stored in an organized way. A fast access to the data is possible by inserting database indexes on certain values. The database generates hashtables for a direct access to the data. This increase in memory accelerates a database query as a complete search through the database is not necessary. Managing the data within a database speeds up the development cycles and makes improvements of any part of the toolchain easier. The database can also be used as common storage place to handle the data access by different computers at the same time. Since a further increase in frequency of one processor has physical limitations, multiprocessors are the way of choice to make processing faster. By using an efficient data structure several cores can work at the same time on one 3D reconstruction. The principle can be further extended to the cloud, enabling the reconstruction of bigger areas in days or hours for which one computer would need several years.

Relational Model

The relational model used in this work, describes the complete database for monocular camera path estimation. The database is designed not only to handle images from one, but from multiple cameras. The images of a stereo camera system or any arbitrary camera setup mounted on a rig can be managed. Also different 3D reconstructions can be handled at the same time in one database. The main idea is the separation of different reconstructions into projects. Two projects can be merged if they contain overlapping areas. A project consists of an `ID`, a `title` naming the object of interest, a detailed `description` of the project and a `date` when it was acquired (see Figure 3.1). Each project consists

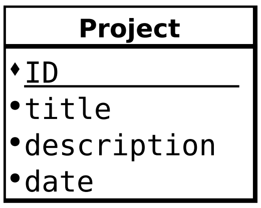


Figure 3.1: A project contains a unique `ID`, a `title`, `description` and an acquisition `date`.

of several images. One image is referenced by a unique `ID`. The database entry of an image consists of a `timestamp`, the `filename` of the actual image, a project ID referencing to the project, a calibration ID and an ID to a position in case the spatial position of the images is already known. Otherwise the position ID will be -1, indicating that the position is not yet known. Most often a limited number of cameras is involved in one project e.g. three cameras when using a trifocal camera setup. Each camera used in a project is designated an ID starting from 0. The set of images taken by one camera is enumerated starting from a frame ID of 0. The frame ID and the camera ID are integrated to improve manual investigations by the user. Only working with an image ID and a calibration ID can be confusing for the user, especially in combination with several projects and multiple cameras. In case of synchronous image acquisition, e.g. with a stereo system or a quadrifocal camera setup (see Figure 8.2), each camera gets its own `cam_ID`. Images that differ in their `cam_ID` but have the same `frame_ID` are taken at the same time. Finally the status of an image is stored in `feature_status`, containing the current state of the feature computation (0: *not computed*, 1: *computing*, 2: *done*). If several computers are working on the same project, each computer will pick an image where the features have not yet been computed. Before each machine begins detecting features and computing the feature description the corresponding feature computation status is set to *computing*. Once a feature list is computed and stored in the database, the status is changed to *done*. The complete data structure for an image is visualized in Figure 3.2. Each image is additionally

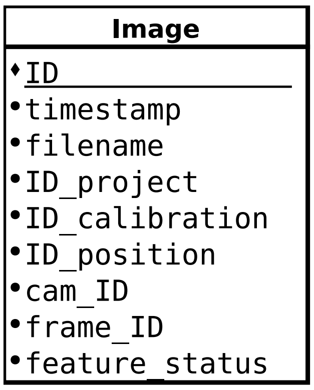


Figure 3.2: The data structure for an image.

linked to a calibration structure (see Figure 3.3) containing the `camera_name`, a `timestamp` and a `description` to store additional information such as aperture and exposure of the camera. Furthermore, the structure contains the parameters belonging to the camera model which were presented in Section 2.2. Knowing from the relational database that several images share the same calibration, enables a restriction in the optimization (Section 5.4.3.1) as the internal camera parameters have to be identical for this set of images.

A feature based approach is used throughout this work (Section 2.3). The underlying data structure is not specified for the exclusive use of SIFT features but a rather general approach for point based features. One feature belongs to an image and is a projection of a 3D point. The latter is modeled even if it is not available yet. All point based feature have an image coordinate (`x`,`y`). `Color`, `scale` and `orientation` can be useful when matching two features. An entry to store the type of description is available, as different types of features use different structures to describe a keypoint. They can be used at the same time, but only features of the same type can be matched against each other. Being able to work with different kinds of features can make the pipeline much more stable. It is more likely that two different types of features will be invented which work successfully in different scenes than, a single one that solves all problems simultaneously. The database structure of a feature is shown in Figure 3.4. If different features were successfully matched, they are mapped to an identical 3D point. They are referenced in a later step to have the same global id. All feature matches are stored in one table. For faster data access not only the feature ID of the correspondences is stored but also the image ID that the feature belongs to. This information might be redundant, but it improves the data access performance significantly at the cost of more memory usage. Inspired by [10], not only the `distance` of the nearest neighbor (NN) in descriptor space is saved but also the `ratio` between the first and the second NN. The second NN has been introduced as the outlier distance [52]. Furthermore, different matching types can be stored as some are more reliable than others (see Figure 3.5). All matches go through different filter steps and the results are stored. By using this strategy it is possible to get back to less reliable matches before running out of matched features.

An important information in all SfM approaches is the relative orientation between two images. This information is computed from a set of matches of two images (Section 5.1). The stored information contains the relative translation as well as the relative rotation. The translation vector $\mathbf{t} = [\mathtt{tx}, \mathtt{ty}, \mathtt{tz}]^T$ is of unit length. The rotation is represented by a unit quaternion $\mathbf{q} = [\mathtt{q1}, \mathtt{q2}, \mathtt{q3}, \mathtt{q4}]^T$.

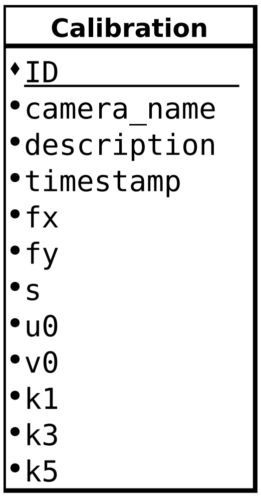


Figure 3.3: Structure containing the calibration of a camera.

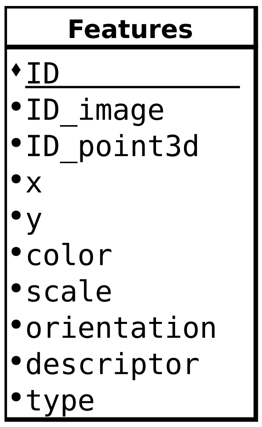


Figure 3.4: Structure of a point based feature.

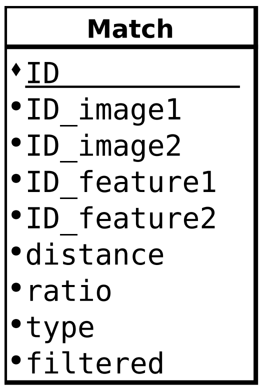


Figure 3.5: A match of two features is stored by their unique feature IDs.

When computing the path of images, redundant information exists as the relative orientation can be computed between all images where at least 5 points are matched. To have an indication which relative orientation is more stable, the depth ratio of the tracked points is stored as well. Since the length of the translation is fixed to 1.0, the mean depth of the triangulated points represents this ratio. The relative orientation is stored in the database as shown in Figure 3.6. At that stage of the processing chain the scaling between two images is not integrated yet. It can be included only when dealing with more than two images. Three images having corresponding features are handled as image triplet. The triplets are extracted from the match table with one SQL query. An image triplet is described by the relative orientation between image 1-2 and image 2-3. Although it is redundant, the relative orientation between image 3-1 is computed as well. If the distance between camera position 1-2 is set to 1.0, the length (`base_scale_1-2-3`) between 2-3 will be computed if common feature points exist. Under the assumption that the relative orientations as well as the scales

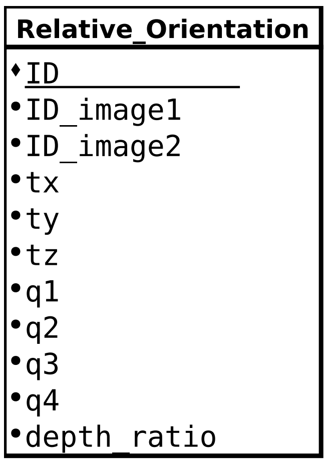


Figure 3.6: Structure to save the relative orientation between two images as well as the depth ratio to the 3D points.

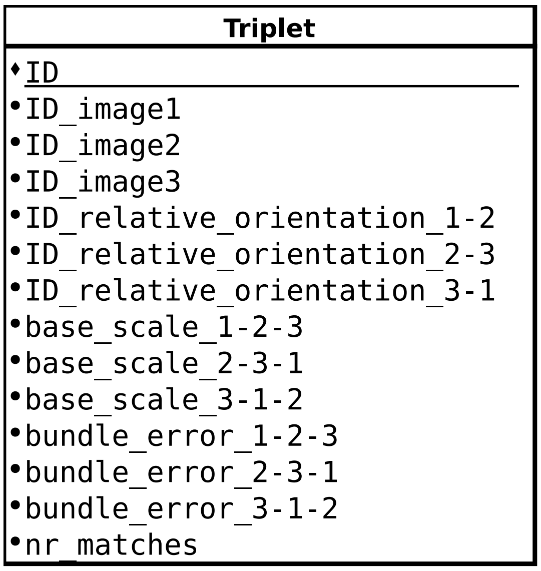


Figure 3.7: The relative orientations and the scales describe an image triplet.

are perfectly computed, the `base_scale_1-2-3` represents the same information as the `base_scale_2-3-1`. Yet, this is not the case in practice. In the second case the relative orientations 2-3 and 3-1 are used to compute the scale. If the computation of one relative orientation fails due to outliers, the system is still able to compute reliable results. Each triplet is optimized with BA as described in Section 5.4.3.1. The resulting reprojection error as well as the number of points are stored enabling a comparison between different triplets regarding their reliability (see Figure 3.7 for the triplet data structure). The complete relational data structure is shown in Figure 3.8.

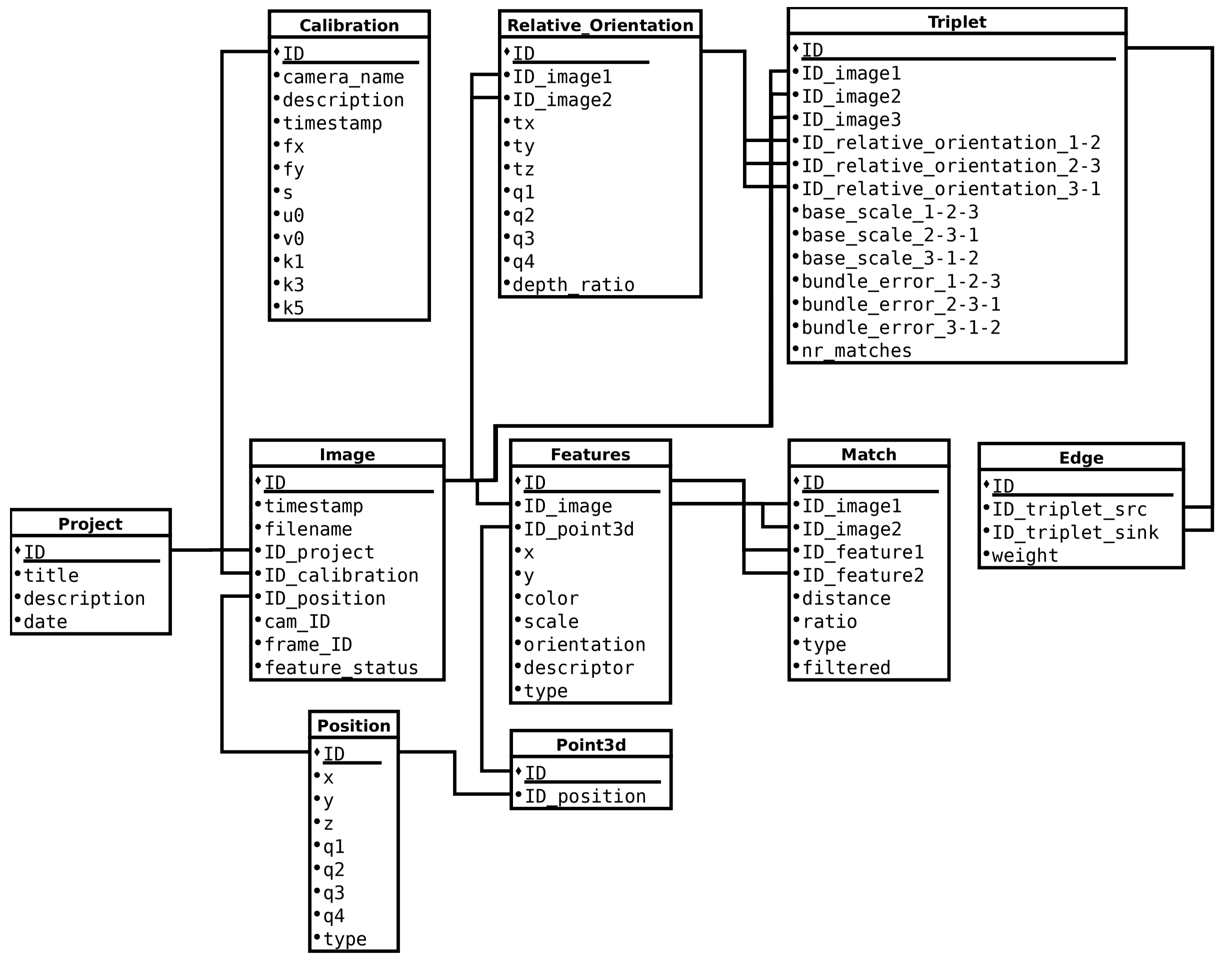


Figure 3.8: Complete relational data structure containing all information necessary to compute a path with a SfM approach.

CHAPTER IV

Matching and Point Identities

Finding correspondences between images can be separated into two domains: matching points and tracking points. Matching points is the general case in which two images show the same content from different perspective but no restrictions are made on the viewpoint difference.

Feature trackers, for instance the Kanade-Lucas-Tomasi (KLT) tracker, rely on an image sequence with a high frame rate e.g. a video [53, 89]. Only small changes between two frames are supposed to exist. Such procedures perform reliably and can be applied in realtime. The major drawback is the drift of points within a video sequence. The point in the initial frame might not be the same as the point in the last frame before the interest point is lost. This drift effect appears in particular in long image sequences. In SfM approaches the balance between a wide and narrow baseline has to be considered. A wide baseline produces in general an accurate 3D point since the intersection angle is larger than for a narrow baseline. In contrast, a narrow baseline simplifies the matching of two images. When a high accuracy is desired, images with a high resolution and a low frame rate are preferred over a video with a high frame rate but low resolution [26].

This work relies on matching images instead of tracking points over a video sequence. The acquired image sequence is sparse in comparison to a video, which makes tracking of feature points infeasible. When considering strong rotations a point can easily move hundreds of pixels between two frames. Nevertheless, a global id has to be assigned to a group of 2D points from several images belonging to an identical 3D point. From a set of pairwise images matches, a global id is propagated from one image point to another (see Section 4.2). These correspondences are used within the optimization described in Section 5.4.3.1. Mismatches can cause inconsistencies, which occur if feature points in one image share an identical global id. The next section describes the classical matching procedure as well as a novel method that allows fast matching. Different filtering

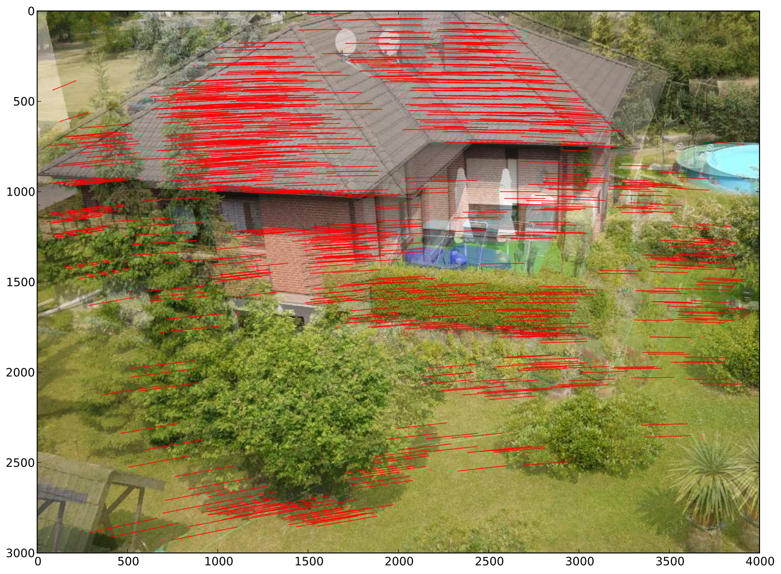


Figure 4.1: Illustration of two images where sparse feature points are matched.

procedures are discussed in Section 4.1.4 to produce a set of outlier free matches.

4.1 Matching

The matching problem can be defined as follows: Given two sets of features the aim is to find a set of tuples where two features of one tuple belong to the same 3D point. As the matching is only based on images, mismatches can occur particularly if repetitive structures are present. Those are often found in man made environments e.g. on facades. Figure 4.1 gives an illustration where two images have been matched to each other. The challenge is to find a large set of true matches while keeping the number of mismatches low, which can mathematically be described as finding the NN in a high dimensional descriptor space. As each feature has a description in $\mathbb{R}^{128}$, finding the NN is time consuming. The naive way of finding the NN is achieved by computing the distance of a descriptor $\mathbf{D}^i_f$, which is the f descriptor of image i, to all descriptors $\mathbf{D}^j$ of image j. A match will be found if the Euclidean distance of the NN is less than a threshold t and the ratio between the best match m_1 and the second best match m_2 is less than r. Comparing against the second best match has been introduced as comparing against the outlier distance [10]. Expressed in formulas:

$$||\mathbf{D}^i_f - \mathbf{D}^j_{m_1}||^2 < t \tag{4.1}$$

$$\frac{||\mathbf{D}_f^i - \mathbf{D}_{m_1}^j||^2}{||\mathbf{D}_f^i - \mathbf{D}_{m_2}^j||^2} < r \tag{4.2}$$

with

$$\mathbf{D}_{m_1}^j = \operatorname*{argmin}_{\mathbf{D}_g^j \in \mathbf{D}^j} \left(||\mathbf{D}_f^i - \mathbf{D}_g^j||^2\right) \tag{4.3}$$

$$\mathbf{D}_{m_2}^j = \operatorname*{argmin}_{\mathbf{D}_g^j \in \mathbf{D}^j \setminus \mathbf{D}_{m_1}^j} \left(||\mathbf{D}_f^i - \mathbf{D}_g^j||^2\right) \ . \tag{4.4}$$

Finding the NN for each point by a linear search has a complexity of $\mathcal{O}(n^2)$ if in both images n points exist. In practice a linear search is too time consuming. Nevertheless, it is still of interest as it is taken as ground truth to evaluate the recognition rate of approximate methods. Many approximate NN approaches have been presented in the past to accelerate the matching [31, 63, 78]. In the next section, the BBF KD-tree matching approach is presented, which is the originally proposed matching scheme by Lowe [52]. A novel more efficient matching procedure, which is named TBH [42], combines the advantages of KD-trees and hash tables.

4.1.1 KD-Tree

NN search in high dimensions has been an ongoing research field. The most common approach is the KD-tree [8], which uses space partitioning in a k-dimensional space to divide the data in equal parts. This binary tree divides the space at each node into halfs. At first, the tree is constructed from the set of all features from one image. Once the tree is created, a query can be applied to find the NN. In case a query point is closer to a hyperplane than to the temporary NN, a backtracking is performed, to find the NN with a high probability. For the original KD-tree, a data point is stored in each node of the tree. It has been shown by [23] that KD-trees provide no speed up for higher dimensions (e.g. $k > 10$). By now several modifications have been developed to increase the speed of KD-trees, which at the end results in an approximate NN approach. [7] describes a modification, named BBF, which was also applied in [52] for the data used in this work. BBF increases the probability that the NN is in a certain bin which is processed before others. This is achieved by storing all backtrackings in a heap sorted by the distance that a query point has to the bin. An approximate NN is found by early cutting off the priority queue. Specially adapted to the data, the next section describes an approach which extends the basic idea of BBF.

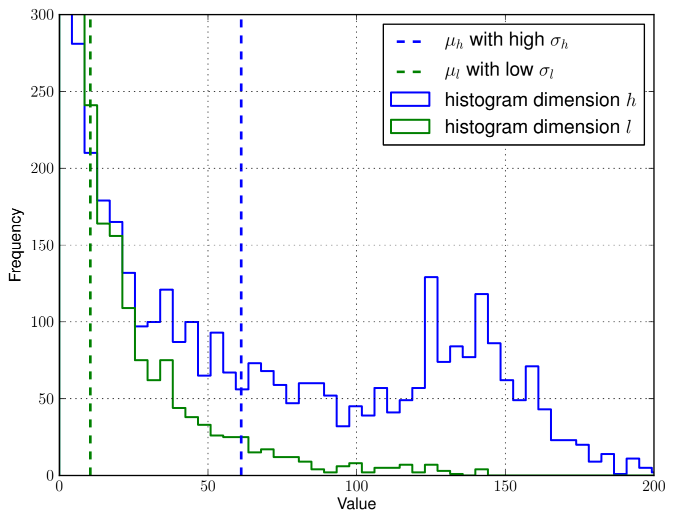


Figure 4.2: Histogram of two different dimensions with high and low variance.

4.1.2 Tree Based Hashing

TBH combines the advantages of a dynamic tree structure with fast direct access of hash tables [42]. Furthermore, TBH is an approach especially developed for the matching scheme described in Equations 4.1 and 4.2. In contrast to the general NN, the search space is restricted to points within a certain radius t of the query point. Since a quadratic distance is used, only points within a maximum distance in one dimension of $\sqrt{t}$ are of interest. As $\sqrt{t}$ can still be large in comparison to the value range 0 to 255 of one dimension, it is assumed that for a true match the difference of two descriptor vectors is spread over several dimensions and not concentrated in one dimension. A strong change in one dimension implies a significant gradient difference in one bin which is not likely for a true match.

Experiments have shown that the error in one dimension is in most cases less than $0.1\sqrt{t}$ for a valid match. Similar to randomized KD-trees [79], the variance of each dimension is evaluated. TBH splits the data in two sets at the mean of the dimension. Using only the dimensions with the highest variance has the advantage that the amount of points within a radius around the mean value is smaller than for a low variance (see Figure 4.2). In case a query point is near the mean value, both left and right child node need to be examined. This procedure is known as backtracking. The amount of points for which backtracking needs to be applied should be minimized, as it degrades the performance. Taking an amount of k dimensions in a decreasing order of their variance, a tree with a maximum of 2^k leaf nodes is generated. In contrast to general KD-trees, all data points are stored in the leaf nodes, so the tree can be converted to a hash table for fast bin access. KD-trees store the data in each node. TBH uses the tree to divide the data in bins. In each bin a linear search is performed. This combination of a search tree and a linear search results in a high precision and great performance.

For all features from the search set, a hash vector is computed. Each entry of the vector contains $\{0, 1\}$. If the index value x_i is less or equal than the mean value of the dimension μ_i, then the hash value h_i is 0, otherwise it is 1.

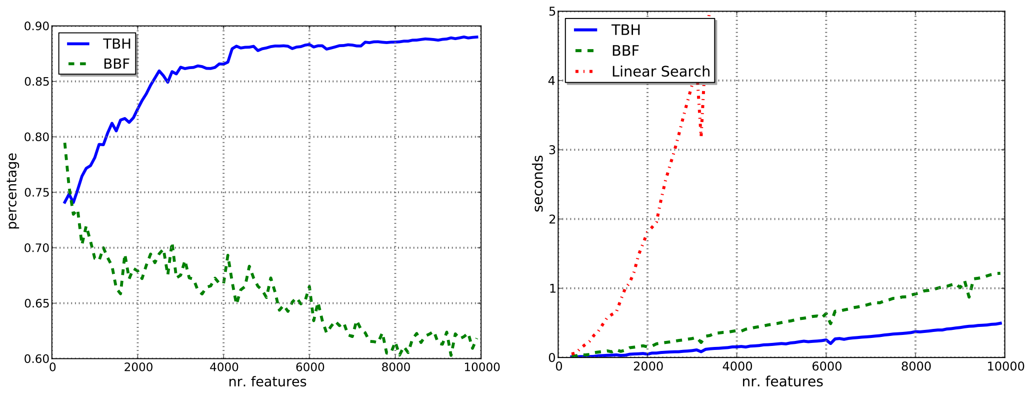
$$h_i = \begin{cases} 0 & : x_i \leq \mu_i \\ 1 & : x_i > \mu_i \end{cases}, i \in [1, k] \tag{4.5}$$

If the value x_i is within a backtracking radius W_i of the mean value μ_i, two hash vectors will be generated. In a tree structure this is referred to as backtracking. A hash value is computed for the first n dimensions, which are sorted by their variance in descending order. For one feature several binary hash vectors will be generated. Each hash vector can be transformed to a decimal value which represents the bin which stores the feature. A maximum amount of 2^k hash vectors will be generated if a backtracking in every dimension is necessary. Filling the hash table is fast, since only k comparisons need to be applied. In contrast to KD-trees, not every subspace is evaluated to select the best dimension to split the data, but each dimensions is evaluated separately. This increases the performance significantly and allows a complete parallel computation of the hash vectors.

The backtracking radius W_i is computed from two parameters. The first parameter τ represents the percentage of features which are allowed to lie within the backtracking radius. With this parameter a high amount of backtrackings can be avoided. Also an upper boundary of a maximum number of backtrackings can be computed, which is important for realtime usage with a deterministic time behavior. The second parameter relates to the threshold $\sqrt{t}$ and is called W_{max}. W_{max} remains for every dimension the same. It is expected that the maximum error is less than 10% of the overall allowed error.

$$W_{max} = 0.1 * \sqrt{t} \tag{4.6}$$

W_i represents the minimum of W_{max} and $W(\tau, i)$, with $W(\tau, i)$ computing a backtracking radius with a constant amount of points for which backtracking is



(a) Precision (b) Performance

Figure 4.3: Precision and performance of TBH in contrast to BBF.

necessary.

$$W_i = \min(W(\tau, i), W_{max}) \tag{4.7}$$

Once all features have been inserted into the hash table, the query set is inserted as well. No backtracking is necessary for the query set, because it has been already performed in the search set. It can be also applied the other way around but the described order of the tasks has the advantage that the correspondences of a query feature need only be searched in one bucket. In each bucket a linear search is performed to find the best and the second best match.

4.1.3 Evaluation

For evaluation the speed of TBH is compared to BBF and linear search. Furthermore, the percentage of matches which overlap with the matches produced by a linear search is evaluated. For both evaluations two images were captured with a small difference in perspective and the correspondence search is performed. The feature lists of both images are truncated to plot the performance and precision over the number of features. Figure 4.3(a) shows the hit rate of TBH and BBF in comparison to a linear search. While the recognition of TBH improves with more features, BBF drops below 65 %. Figure 4.3(b) shows performance of TBH, BBF and linear search. As the matching time of linear search increases quadratically with the number of feature, it is not suitable for real applications. The execution time of TBH is twice as fast as BBF with a better precision. Adjusting the hit rate of BBF by adding more dimensions results in an increased execution time. The hit rate of TBH is remarkably high with up to 90 %. By increasing the amount of backtrackings, even a hit rate of 95 % can be achieved

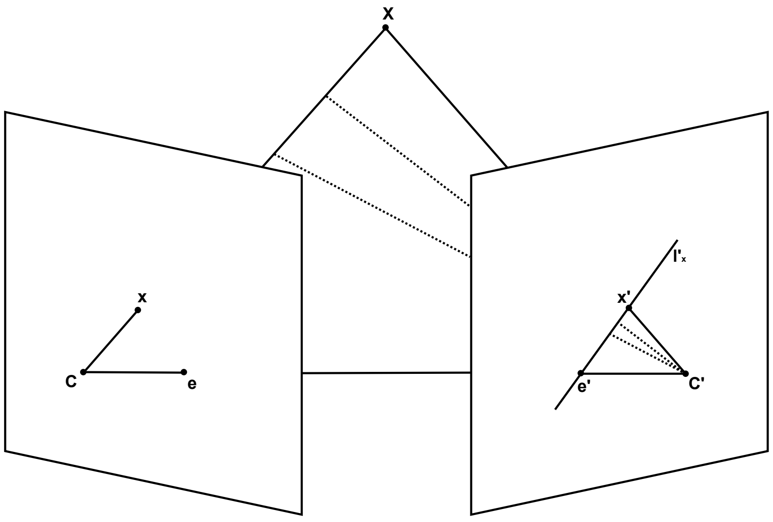


Figure 4.4: Epipolar geometry of two views.

but at the expense of speed.

4.1.4 Filtering

Filtering mismatches is an essential part for robust 3D reconstruction. As outliers are hard to model, a set of matches containing almost no outliers is necessary for a robust 3D reconstruction toolchain. In this section, three outlier filters are presented. The first filter is based on the epipolar geometry of two images, while the second approach filters using a trifocal tensor. The third filter uses the reprojection error after a BA step.

4.1.4.1 Epipolar Filter

The epipolar geometry generally describes the setup between two perspective views and a point in space. Figure 4.4 gives an illustration of two image planes and a 3D point. The point $\mathbf{X}$ is projected into a camera C as point $\mathbf{x}$ and in a second camera C' as point $\mathbf{x}'$. Knowing the relations between the two image planes, a line $\mathbf{l}'_{\mathbf{x}}$ can be computed where the corresponding point $\mathbf{x}'$ must lie on and vice versa. This relation can be computed by knowing a set of at least 7 correspondences and is described by the fundamental matrix $\mathbf{F}$. The following constraint has to hold for each valid image correspondence $\mathbf{x}$ and $\mathbf{x}'$ also known as the epipolar equation.

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0 \tag{4.8}$$

The rank of $\mathbf{F}$ is 2, which means $\det(\mathbf{F}) = 0$. Given $\mathbf{F}$, the line in the second view can be computed by $\mathbf{l}'_{\mathbf{x}} = \mathbf{F}\mathbf{x}$. As the second point is fixed to a line, this relation is used to detect outliers.

Before being able to reject outliers, the fundamental matrix is computed with the well known normalized 8 point algorithm [28]. As the initial set of correspondences computed in Section 4.1.2 still contains outliers, a RANSAC approach is used for a robust estimation. Once the fundamental matrix has been determined, all matches are evaluated using the Sampson distance. The Sampson distance is the first-order approximation of the geometric error [27]. The geometric error is important for real world applications, as the algebraic error is only a mathematical construct. The error can be computed by triangulating $\mathbf{x}$ and $\mathbf{x}'$ and back projecting the resulting 3D point $\mathbf{X}$ in the two images. The difference between the back projected 2D point and the actual measurement is the geometric error. Since this procedure is time consuming, the Sampson error is used in practice. In case of the fundamental matrix, the error ϵ of a point i is given by

$$\epsilon = \frac{\mathbf{x}_i'^T \mathbf{F}\mathbf{x}_i}{(\mathbf{F}\mathbf{x}_i)_1^2 + (\mathbf{F}\mathbf{x}_i)_2^2 + (\mathbf{F}^T\mathbf{x}_i')_1^2 + (\mathbf{F}^T\mathbf{x}_i')_2^2} \; . \tag{4.9}$$

The subindex j of the bracket $(..)_j^2$ refers to the j^{th} entry of the containing vector. In case ϵ is higher than a predefined threshold, the match is rejected. Otherwise the matches are stored in the database with the structure presented in Figure 3.5.

4.1.4.2 Trifocal Filter

As outliers can still pass the epipolar filter if the selected point lies on the epipolar line, an additional filter step can be applied for image triplets. The trifocal tensor is utilized to estimate the projective relation between three views. With the underlying data structure, one can efficiently acquire matches which coincide with three images. When a feature from image one is matched with image two and three, the corresponding features from image two and three have to match as well. Having a list of triplet matches, the trifocal tensor $\mathbf{T}$ is computed in a robust manner following the approach presented in [90]. The trifocal tensor is a $3 \times 3 \times 3$ tensor describing the trilinear relations. Knowing three corresponding points $(\mathbf{x}, \mathbf{x}', \mathbf{x}'')$, they must fulfill

$$[\mathbf{x}']_\times \left(\sum_i \mathbf{x}_i \mathbf{T}_i \right) [\mathbf{x}'']_\times = \mathbf{0}_{3\times 3} \; . \tag{4.10}$$

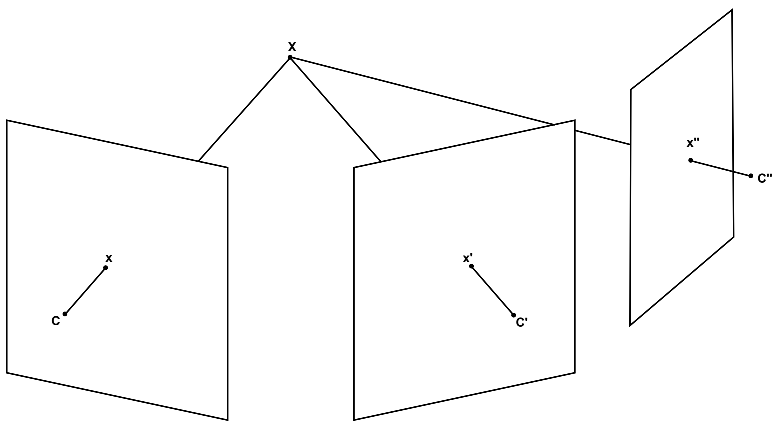


Figure 4.5: Camera setup illustrating the trifocal tensor with point correspondences.

From $\mathbf{T}$, three projection matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ are created. A 3D point $\mathbf{X}$ is computed for each triplet $(\mathbf{x}, \mathbf{x}', \mathbf{x}'')$ (see Figure 4.5). The maximum reprojection error of each triplet decides whether the triplet passes the filter or is rejected in further computational steps.

4.1.4.3 Filter after Bundle Adjustment

In contrast to the trifocal filter, a metric triplet is computed in the processing chain. A metric triplet consists of two relative orientations and one scaling factor. Each triplet is optimized by a BA. The optimization refines the camera parameters as well as the 3D points. As presented in Section 2.4.3, the optimization has a robust cost function to reduce the influence of outliers. After the optimization, each point $\mathbf{X}_i$ is projected in each camera j. If the reprojection error in one camera is higher than a threshold ϵ, the corresponding matches will not pass the filter.

$$\max_j ||\mathbf{P}_j \mathbf{X}_i - \mathbf{x}_{i,j}|| < \epsilon \tag{4.11}$$

There are many triplets with overlapping matches, since one image pair can belong to many different triplets. A filter strategy is applied in which each triplet can verify but not reject a match. Otherwise matches could be rejected by wrong computed triplets. The filter is only applied to triplets which could be successfully bundled. A triplet is accepted if the mean reprojection error is low (less than 2 pixel) and the optimized scale and relative orientation is within a certain range of the initial input. Otherwise, it is assumed that the BA solution is only a local minimum. If the maximum error of the reprojection of point i is less than ϵ, the entries in the database which form the matches are marked as passing the filter (see Figure 3.5). Thus, a match can still be marked as passing the filter when it is part in other triplets.

4.2 Point Identities

Tracking describes the procedure of following a feature over an image sequence. It is essential that the feature always belongs to the same point in space and does not drift away over time. The most popular tracking approach is the KLT-Tracker [53, 89, 77], which has been shown to work well with video sequences. In this work local features are more suitable, because consecutive frames can look significantly different. Nevertheless, pairwise matches between features have to be grouped to assign a unique id to corresponding features. This is necessary for the final optimization step described in Section 5.4.3.1. As this optimization can only handle outliers to a certain degree, a correct id propagation is essential.
A set of feature pairs of several image combinations serve as the input to id propagation. The set follows the structure presented in Figure 3.5. At first the global ids of all features belonging to the project are set to 0. For each image, iterating over all images according to the order of the frame id, a set of features is selected which has predecessor matches which passed the filter. This procedure results in features which exhibit matches to images with a lower frame id. To avoid mislabeling, only the features, where the corresponding match is marked as filtered, are selected. If different filtering approaches are available, it will be recommended to start with matches which passed the most reliable filter. In case no more matches are found, going back to less reliable filter approaches or even assigning ids to matches which did only pass the epipolar filter is possible. As most features have more than one predecessor, it can occur, that ids of the predecessors are different. There are two possibilities to solve this situation. The first method merges the different ids to a unique id, which can result in gross errors in case of mismatches. The second more reliable method computes the id which is most probable. The id is assigned following a decision by majority to keep the computation time low. In case there is no predecessor with a global id different to zero, a new id is generated and assigned to all predecessors as well as to the current feature. If an inconsistency is detected, e.g. two features in one image share an identical global id, the global id of all corresponding features is reset to 0. For a more programming oriented description see Algorithm 1.

Algorithm 1 Assign global ids to features.

```
for all $im$ in images do
  for all $feature$ in $im$ do
    if $feature$ has filtered predecessors then
      $preFeatures$ = getPredecessors($feature$)
      $id$=getIdWithHighestProb($preFeatures$)
      if $id > 0$ then
        assign $id$ to $feature$
      else
        assign new id to $feature$ and all $preFeatures$
      end if
    end if
  end for
  checkAndCorrectInconsistency($im$)
end for
```

CHAPTER V

Multi View Geometry

This part of the thesis describes the computation of the pose (translation and rotation) of each image which will be computed relative to a reference coordinate system. In computer vision this task is known as SfM. In robotics the same problem is addressed under the term SLAM [12]. In SfM approaches the problem can be divided in computing the relative orientation between two images and selecting the scale between their camera positions. For two images the scale is ambiguous. For three images a scale can be computed e.g. in a triplet between image two and three. This scale is relative to the scale which was arbitrarily chosen between camera position one and two. The first two camera poses commonly define the reference coordinate system. The first camera is placed at the origin. If no prior knowledge is given (through markers or ground control points (GCP)) the distance between the first and second camera position will be set to a known metric distance or simply to 1.0. While moving through the scene, the errors that occur when concatenating the relative orientations and propagating the scale will accumulate with every new image. An essential task is the reduction of the accumulated error in an optimization step. If the initial computation of the path is unspecific due to the accumulation of large errors, it is likely that the non-linear optimization will only find a local minimum. A good initial estimate is needed in order to find the satisfying solution. A graph-based method to estimate a reliable initial path by propagating the scale over image pairs and triplets, for which parameters are estimated with a high accuracy, is presented in the following sections. As it is not possible to select a scale separately for an image pair without prior knowledge, the presented algorithm is based on image/camera triplets. A score is assigned to each triplet based on an empirically evaluated cost function.

5.1 Relative Orientation

The relative pose of two images can be described by the essential matrix. It holds the following condition based on the epipolar geometry where $\mathbf{q}$ and $\mathbf{q}'$ are normalized image coordinates of a homogeneous point $\mathbf{p}$ from one view and $\mathbf{p}'$ the corresponding point in a second view. The definition of normalized image coordinates is given by

$$\mathbf{q} = \mathbf{K}^{-1}\mathbf{p}\ . \tag{5.1}$$

The essential matrix is analogous to the fundamental matrix, but for the calibrated case, which is directly computed from normalized image coordinates.

$$(\mathbf{q}')^T\mathbf{K}'^T\mathbf{F}\mathbf{K}\mathbf{q} = 0 \tag{5.2}$$

$$(\mathbf{q}')^T\mathbf{E}\mathbf{q} = 0 \tag{5.3}$$

Various algorithms have been described previously to compute the essential matrix from a list of point pairs. In all cases the internal calibration of the cameras has to be known and a minimum of 5 point pairs is needed to compute the essential matrix [65]. The essential matrix is a 3×3 matrix, which holds the following conditions:

$$\mathbf{E} = \mathbf{R}[\mathbf{t}]_\times \tag{5.4}$$

where $[\mathbf{t}]_\times$ is the skew-symmetric matrix of the translation vector $\mathbf{t}$ and $\mathbf{R}$ is the rotation between the first and second camera. The decomposition of $\mathbf{E}$ is calculated using Singular Value Decomposition (SVD)[28]

$$\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{5.5}$$

with $\mathbf{U}$ and $\mathbf{V}$ being orthogonal matrices and $\mathbf{D}$ is a diagonal matrix.

$$\mathbf{D} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{5.6}$$

Since the skew-symmetric matrix in Equation 5.4 has rank 2, the essential matrix has as well a rank of 2, so the last diagonal element of $\mathbf{D}$ is zero. Additionally, the two nonzero eigenvalues have to be equal. The rotation $\mathbf{R}$ has two algebraically valid solutions which can be composed by

$$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^T \tag{5.7}$$

and

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^T\mathbf{V}^T \tag{5.8}$$

with

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.9}$$

The translation vector $\mathbf{t}$ is the eigenvector belonging to the eigenvalue zero in $\mathbf{D}$, which is the last/third column of $\mathbf{U}$. The scale of $\mathbf{t}$ is ambiguous. Usually the vector is normalized to a length of 1.0.

$$\mathbf{t} \sim \mathbf{U}[:, \mathbf{2}] \tag{5.10}$$

With $\mathbf{t}$, $-\mathbf{t}$ and two rotation matrices, four possible solutions exist, but only one solution is valid regarding the underlying camera geometry. The cheirality constraint enforces the triangulated point to be in front of both cameras, so the right solution can be easily identified [28].

Different algorithms to estimate the essential matrix have been presented in literature [46, 65, 71, 28]. Many algorithms have problems with critical motions or degenerate point configurations. A detailed analysis of those algorithms can be found in [72, 74]. The five-point pose estimator [65] has been proven to produce reliable results even under certain critical configurations. A minimal parametrization is always preferable. The section proceeds as follows. A method to select well distributed points is described before reviewing the five-point algorithm. The outcome of this algorithm serves as input for the newly developed CCHC approach which replaces RANSAC in the toolchain presented in this thesis.

5.1.1 Selecting Distributed Points

To compute a reliable relative orientation it is essential to select well distributed points, otherwise the result can be wrong or unstable. Since five points are needed to compute the essential matrix, a well distributed set of five point pairs should to be selected to compute a precise relative orientation. The first point is selected randomly with a uniform distribution. After the first point

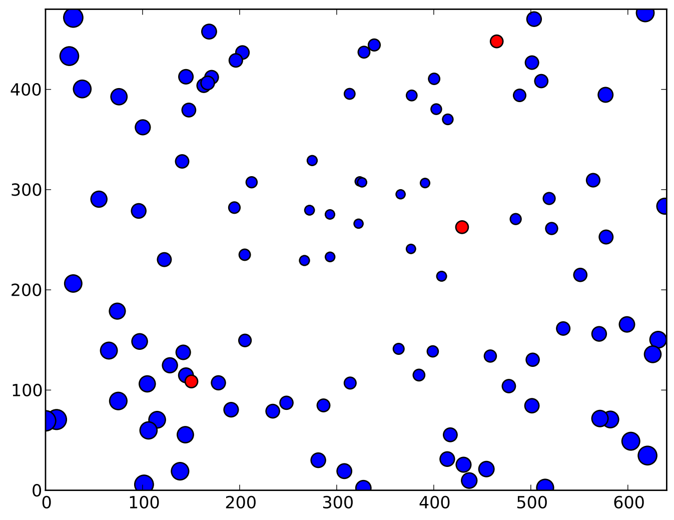


Figure 5.1: Selectable points in blue with the radius representing the weight, already chosen points in red.

was added to the set of chosen points, all other points are weighted according to their squared summed distance to already chosen points. In each iteration a new point is selected randomly from a distribution which is generated considering all weights. Assuming a number of n points and m selected points, the weight for each point is computed by

$$w_i = \sum_{j=1}^{m} ||\mathbf{p}_i - \mathbf{q}_j||^2 \tag{5.11}$$

for $m > 0$ where $\mathbf{q}_j$ is an already selected point and $\mathbf{p}_i$ is a candidate. To select a new point according to the weights, an accumulated weight is computed:

$$c_i = \sum_{j=1}^{i} \frac{w_j}{\sum\limits_{k=1}^{n} w_k} \tag{5.12}$$

A random number $r \in [0, 1)$ is taken to select the index of a new point.

$$index = \min i \,|c_i - r > 0 \tag{5.13}$$

This point is removed from the list of candidates and inserted in the list of selected points. This procedure is repeated until the amount of five selected points has been reached. An intermediate result, in which three points were chosen (red) and two points need to be selected, is shown in Figure 5.1. Additionally, the weights of the candidates (blue) represented by their radius are displayed.

5.1.2 Five-Point Algorithm

This section follows the approach presented in [65, 83]. As already shown by Kruppa in 1913 [40], the relative orientation between two calibrated images can be computed by a minimum of 5 point pairs [86]. To solve the relative orientation, the roots of a tenth degree polynomial need to be computed [65]. This special case can be solved in closed form with the help of Gröbner basis [83]. Since the essential matrix has rank two the following equation holds:

$$det(\mathbf{E}) = 0\,. \tag{5.14}$$

Starting from Equation 5.6 the following cubic constraint is formulated since the two non-zero eigenvalues are equal:

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0\,. \tag{5.15}$$

At first Equation 5.3 can be rewritten as a scalar product.

$$\tilde{\mathbf{q}}^T\tilde{\mathbf{E}} = 0 \tag{5.16}$$

with

$$\tilde{\mathbf{q}} = [\mathbf{q}_1\mathbf{q}'_1, \mathbf{q}_2\mathbf{q}'_1, \mathbf{q}_3\mathbf{q}'_1, \mathbf{q}_1\mathbf{q}'_2, \mathbf{q}_2\mathbf{q}'_2, \mathbf{q}_3\mathbf{q}'_2, \mathbf{q}_1\mathbf{q}'_3, \mathbf{q}_2\mathbf{q}'_3, \mathbf{q}_3\mathbf{q}'_3]^T \tag{5.17}$$

A vector $\tilde{\mathbf{q}}$ exists for each point pair, resulting in a total number of five vectors. The scalar product of each vector is combined in a design matrix $\mathbf{A}$ with a size of 5×9.

$$\mathbf{A}\tilde{\mathbf{E}} = 0 \tag{5.18}$$

Since $\mathbf{A}$ has not full rank, applying a SVD to $\mathbf{A}^T\mathbf{A}$ results in four eigenvalues equal to zero. Defining the vectors forming the null space as $\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}, \tilde{\mathbf{Z}}, \tilde{\mathbf{W}}$, the essential matrix is given by

$$\mathbf{E} = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z} + w\mathbf{W} \tag{5.19}$$

with x, y, z, w being unknown scalars. As the essential matrix can be multiplied by an arbitrary scalar and still represents the same camera configuration, one parameter can be chosen arbitrarily e.g. $w = 1$. From Equations 5.14 and 5.15 it is possible to formulate ten third-order polynomial equations with unknowns x, y, z. The equation system can be either solved by Gauss-Jordan elimination [65] or with the help of Gröbner basis [16, 83]. A maximum of ten real solutions can exist. The geometrically correct solution can be selected by taking at least one additional point into account. Both [65, 83] suggest to use RANSAC in an additional step to select the solution which is supported by most of the other point pairs. Especially in the case of critical configurations, RANSAC often selects a degenerate configuration [21]. The next section presents a Hierarchical Clustering (HC) approach as a better alternative to RANSAC for computing the relative orientation.

5.1.3 Hierarchical Clustering

In man made environments, the majority of points may lie on a plane, which is a critical configuration for many algorithms such as the 8-point algorithm [72]. In contrast, the 5-point algorithm is stable to planar configurations [65]. In general, a configuration of two cameras and 3D points is critical if all points and the camera centers lie on a ruled quadric [34]. In practice, quasi-degenerate configurations appear. In this case most of the points lie on a critical surface and only a small subset of points is not in a critical position. It is most important to include these points in the computation of a correct relative orientation. Often these points are mistakenly identified as outliers. A standard RANSAC approach seems to prefer such degenerate configurations [21]. RANSAC assumes that a potential true solution was found because of a high number of inliers which all belong to a degenerate surface. But even if a correct solution was in the set of drawn samples, RANSAC often selects a degenerate solution. The degenerate solution better fits the data, resulting in a higher number of inliers. For the true solution many inliers could possibly be marked as outliers due to noise. Defining an outlier threshold can be complicated as the statistics of the data is not known. Degenerate configurations appear quite often in real world data. Images of houses, where most of the interest points lie on a flat surface, are a prominent example.

A modified HC approach is presented here which is able to handle quasi-degenerate configurations robustly and efficiently. In comparison to RANSAC, which finds the consensus in the input space, the here presented HC approach additionally establishes the consensus in solution space. A distance measurement in the underlying space has to be defined to be able to apply HC.

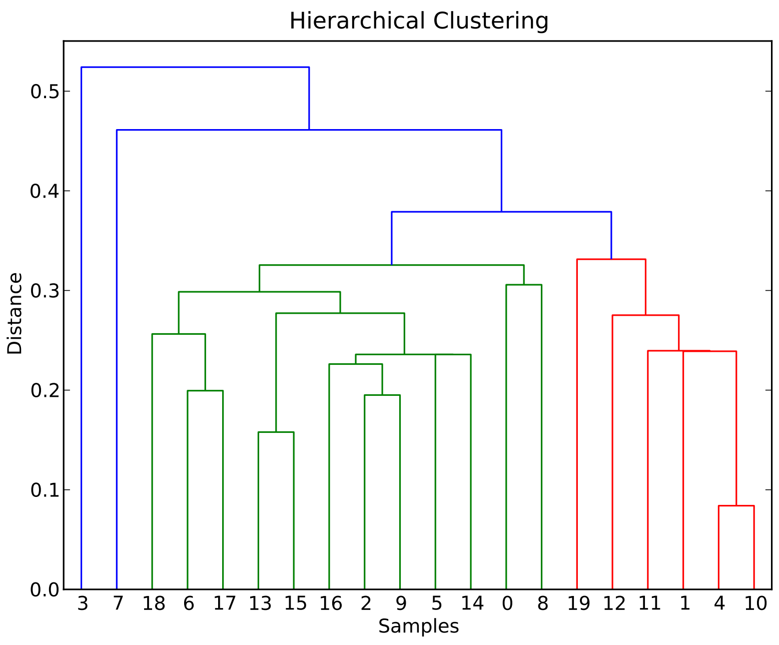


Figure 5.2: General hierarchical clustering scheme.

In case of the relative orientation, the motion belongs to a Lie Group. With the corresponding Lie Algebra, the distance measurement is taken from the associated vector space. Similarly, Subbarao et al. [87] applied a mean shift approach to reliably find the relative pose of two cameras. In their case, they assume that the relative pose between consecutive frames only changes by a small amount. A prior solution is taken as a starting point for mean shift clustering.

The presented approach handles the general case, where only two images with putative point correspondences are given. HC belongs to the group of agglomerative clustering methods which build the clusters bottom up. At the beginning each data point (here the relative orientation) forms a single cluster. In each iteration the two closest clusters are merged until only one cluster is left or the cost of merging two clusters is too high. Figure 5.2 shows a general HC scheme.

Within HC, a linkage criterion between two clusters is needed. The criterion is based on a certain distance between the entries of two clusters. Considering rotations and translations, several distance measurements are possible. A rotation in 3D space belongs to the special orthogonal group $SO(3)$ which is a subgroup of a general linear group $GL(3)$ with the following properties:

$$SO(3) = \{\mathbf{R} \in GL(3) \mid \mathbf{R}\mathbf{R}^T = \mathbf{I}\ ,\ det(\mathbf{R}) = 1\} \tag{5.20}$$

An invertible transformation of homogeneous coordinates in 3D belongs to the group $GL(4)$. Since a relative orientation is a special form of the $GL(4)$ it is represented by the subgroup $SE(3)$.

$$SE(3) = \left\{ \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} \in GL(4) \mid \mathbf{R} \in SO(3) \ , \ \mathbf{t} \in \mathbb{R}^3 \right\} \tag{5.21}$$

The Lie group $SE(3)$ has a corresponding algebra $\mathfrak{se}(3)$ which is the tangent space. As the special Euclidean group $SE(3)$ is not a vector space and has no distance measurement, the correspondence between the Lie group $SE(3)$ and the Lie algebra $\mathfrak{se}(3)$ is established by an exponential mapping

$$exp : \mathfrak{se}(3) \rightarrow SE(3) \tag{5.22}$$

and vice-versa

$$log : SE(3) \rightarrow \mathfrak{se}(3) \ . \tag{5.23}$$

In the tangent space the distance measurement is defined. A more detailed description how to present an Euclidean motion as a lie algebra can be found in [3].
In the presented approach the distance measurement is simplified to the relative translation. In many cases the relative rotation is well established but when only a small parallax exists, the relative translation can be wrong. This results in a much faster solution with only a small loss of accuracy.

5.1.3.1 Cross Compare Hierarchical Clustering

Given a set of n relative orientations and their corresponding five point pairs from which each relative orientation was computed, a new HC approach is proposed. This modified HC is named CCHC. First, a distance matrix containing the distances between each relative orientation is computed. In the conventional HC approach, the two clusters which have the closest distance are selected and merged to a bigger cluster. The distance matrix is updated according to the new cluster. Recomputing the distance of the new cluster to all other clusters can be time consuming.

This work proposes a winner-takes-it-all merging procedure. In each merging step one cluster (sample) survives. No new distances need to be computed. The cluster which survives is defined by a strategy termed cross compare. A cross similarity between two relative orientations is computed to define which relative orientation is superior. The cross compare is twofold. In a first step the number of inliers are counted in a cross compare manner.

The inlier count of an essential matrix $\mathbf{E}_A$, computed from points $\mathbf{p}_i, \mathbf{p}'_i$, is evaluated with the points from which $\mathbf{E}_B$ $(\mathbf{q}_i, \mathbf{q}'_i)$ is computed and vice versa. In a second step, the Sampson error is computed also in the cross compare manner, if both relative orientation have the same amount of inliers.

$$\epsilon_A = \sum_{\forall i} \frac{\mathbf{q}'^T_i \mathbf{E}_A \mathbf{q}_i}{(\mathbf{E}_A \mathbf{q}_i)_1^2 + (\mathbf{E}_A \mathbf{q}_i)_2^2 + (\mathbf{E}_A^T \mathbf{q}'_i)_1^2 + (\mathbf{E}_A^T \mathbf{q}'_i)_2^2} \tag{5.24}$$

$$\epsilon_B = \sum_{\forall i} \frac{\mathbf{p}'^T_i \mathbf{E}_B \mathbf{p}_i}{(\mathbf{E}_B \mathbf{p}_i)_1^2 + (\mathbf{E}_B \mathbf{p}_i)_2^2 + (\mathbf{E}_B^T \mathbf{p}'_i)_1^2 + (\mathbf{E}_B^T \mathbf{p}'_i)_2^2} \tag{5.25}$$

Assuming $\mathbf{E}_A$ is the correct solution and the points $\mathbf{q}$ belong to a degenerate configuration, the points $\mathbf{q}$ nevertheless support the correct solution, whereas not all points $\mathbf{p}$ will support the degenerate solution. In this case A has more inliers than B, the relative orientation A is superior to B and wins the merge. If there are outliers within the point samples $\mathbf{q}$, only a partial amount of the samples will support $\mathbf{E}_A$ but hardly any points $\mathbf{p}$ will support $\mathbf{E}_B$, as it represents a false estimate produced by outliers. Again A has more inliers than B and the relative orientation A wins the merge. In case the number of inliers are the same, ϵ_A and ϵ_B decide which relative orientation is superior. When only one cluster is left, the relative orientation which survived most merges is selected as solution. In most cases, this cluster is identical with the final cluster. A programming oriented description is given in Algorithm 2.

5.1.3.2 Good Sample Probability

The main parameter within CCHC is the number of relative orientations that are computed. In RANSAC, the probability to draw no outliers is

$$p_{no} = (1 - \epsilon_o)^s \tag{5.26}$$

with s being the number of points which are necessary to compute the model and ϵ_o is the percentage of outliers. The probability that a set of n models contains at least one model which is computed from a good point set is

$$p = 1 - (1 - p_{no})^n \ . \tag{5.27}$$

Algorithm 2 Choosing a relative orientation from multiple samples

```
reloriVector=computeMultipleRelativeOrientations()
nrValid=reloriVector.size() //number of remaining clusters
$\mathbf{C}$ = computeCostMatrix(reloriVector)
while nrValid>1 do
  i,j=getMinInd($\mathbf{C}$)
  $in_1$,$in_2$=crossInlier(reloriVector[i],reloriVector[j])
  if $in_1 == in_2$ then
    $\epsilon_1$,$\epsilon_2$=crossSimilarity(reloriVector[i],reloriVector[j])
  else
    $\epsilon_1$=$in_2$
    $\epsilon_2$=$in_1$
  end if
  if $\epsilon_1 < \epsilon_2$ then
    $\mathbf{C}$[j,:]=$\infty$
    $\mathbf{C}$[:,j]=$\infty$
    incSurvivedMerges(reloriVector[i])
  else
    $\mathbf{C}$[i,:]=$\infty$
    $\mathbf{C}$[:,i]=$\infty$
    incSurvivedMerges(reloriVector[j])
  end if
  nrValid=nrValid-1
end while
winInd=getIndexMaxSurvived(reloriVector)
return reloriVector[winInd]
```

For CCHC, Equation 5.27 needs to be changed to the probability mass function of the binomial distribution to ensure a cluster size of at least c.

$$p = 1 - \sum_{k=0}^{c-1} \binom{n}{k} p_{no}^{k} (1 - p_{no})^{n-k} \tag{5.28}$$

Generalizing Equation 5.28, with a probability to draw a good sample p_{gs}, leads to

$$p = 1 - \sum_{k=0}^{c-1} \binom{n}{k} p_{gs}^{k} (1 - p_{gs})^{n-k} \ . \tag{5.29}$$

Under the circumstances of outliers and points on a degenerate surface, the probability to draw a good sample is the product of p_{no} and p_{nd}.

$$p_{gs} = p_{no} p_{nd} \tag{5.30}$$

If at least m points need to be drawn from the non-degenerate set of points to computed a valid model, the probability p_{nd} is computed as

$$p_{nd} = \sum_{i=0}^{s-m} \binom{s}{i} \epsilon_d^{i} (1 - \epsilon_d)^{s-i} \tag{5.31}$$

with ϵ_d being the percentage of degenerate points.

5.1.3.3 Evaluation CCHC vs. RANSAC

This section compares CCHC with RANSAC for the 5- and 8-point algorithm. A simulation, in which most 3D points lie on a plane and only a small percentage is spread arbitrarily in space, is used to compare these two approaches. Since a plane is not a degenerate configuration for the 5-point algorithm [83], the 8-point algorithm is used for the evaluation also. Two virtual cameras are placed randomly in a box of size 1 × 1 × 1 and with an offset of -2 in z-direction, looking towards the center (see red box in Figure 5.3). For each pair of virtual cameras, the projection of the 3D points is used to calculate a set of 1000 relative orientations with the 5-point algorithm presented in Section 5.1.2 with well distributed points as presented in Section 5.1.1. As this planar configuration is not critical for the 5-point algorithm, the simulation is repeated with relative orientations computed from the 8-point algorithm. The probability to compute at least 20 good samples is higher than 99 % for each evaluated configuration.

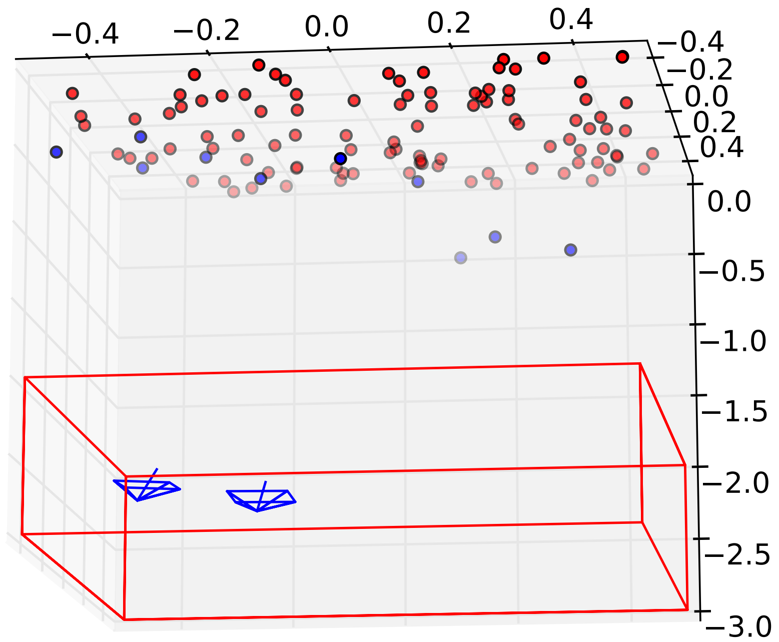


Figure 5.3: Top view of the simulation setup with degenerate points in red.

The probabilities are computed as discussed in Section 5.1.3.2. From this set of relative orientations the one which is closest to the true camera configuration is to be chosen. The accuracy of CCHC and RANSAC is computed for different amounts of Gaussian noise which is added to the projection of the 3D points. The accuracy is also evaluated with different percentages of outliers. For evaluating the outlier behavior the amount of Gaussian noise is fixed to $\sigma = 2.0$. The number of points (blue) which does not lie on the degenerate surface is varied from 10 - 80 points, while 100 points lie on a plane. For each combination (noise / outlier and number of points) the simulation is repeated 1000 times. The outlier threshold within RANSAC and CCHC is set to $3 * 2$ pixel. The error is measured as the minimum angle between the true translation vector and the estimated one. The mean absolute error (MAE) as well as the root mean square error (RMSE) between the estimated and the true solution are shown in Figures 5.5 and 5.6 with no outliers. The performance of both approaches decreases with increasing amount of noise.

For the 5-point algorithm, the slope of RANSAC is much larger than the slope of CCHC. The performance of RANSAC is similar for different number of points in front of the plane. CCHC is more precise for the quasi-planar case. The influence of noise decreases for CCHC for more than 60 point in front of the plane (see Figure 5.5).

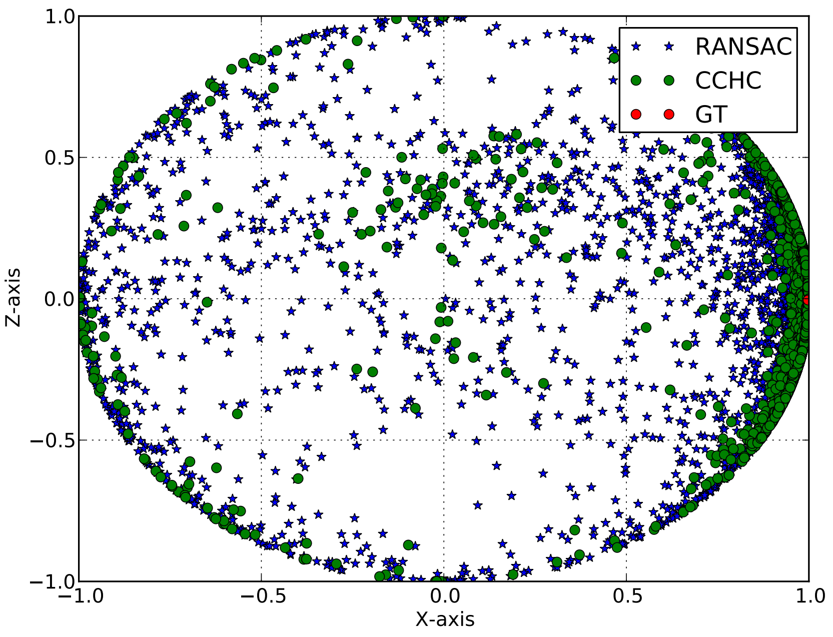


Figure 5.4: Distribution of the relative translation.

For the 8-point algorithm, the behavior is different. CCHC performs much better than RANSAC if only 10 points are in front of the plane. CCHC shows to be more stable to quasi-degenerate configurations. The performance of RANSAC and CCHC increases as more points are in front of the plane. For the non-degenerate configuration both approaches perform similar (see Figure 5.6).

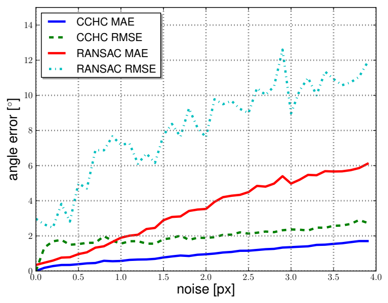
Figures 5.7 and 5.8 show the performance under the influence of outlier. The results produced by CCHC are at least as good as the results from RANSAC. Most of the time the results are better. The density of the cluster center has a big influence about the results of CCHC. For this reason the improvement with CCHC is much larger for the 5-point algorithm than for the 8-point algorithm. Please note that the range of the y-axis differs for the 5-point algorithm and the 8-point algorithm

In addition to simulated data, CCHC and RANSAC are evaluated on real world data. The stereo setup from the car dataset (see Section 7.4) is chosen to evaluate the relative orientation. In this setup, the orientation of the camera pair remains constant whereas the scene changes while driving around. Due to the spherical lense of the GoPro HD HERO 2 camera the appearance of the object in both images changes a lot, especially at the margin of the images. In total 5000 images from a video sequence were extracted, each stereo pair is matched, and the relative orientation is computed. For 4954 out of 5000 images the computation of

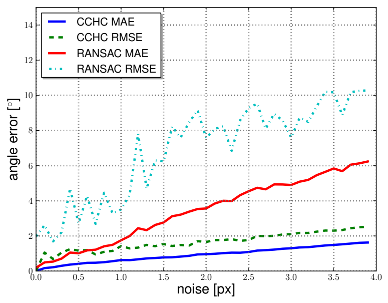
CCHC	RANSAC
0.37	0.77

Table 5.1: RMSE of the relative translation difference to ground truth.

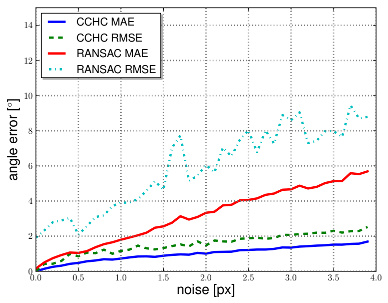
the relative orientation was successful. The computation failed for the remaining pairs due to an insufficient number of correct matches. Once again, for each pair 1000 relative orientations were computed, which served as input for CCHC and RANSAC. The results were compared to ground truth, which has been computed from manually selected image pairs which were free of outliers and optimized by BA. Figure 5.4 shows the distribution of the relative orientation projected onto the X-Z plane. The samples produced by RANSAC show a wide spread over the complete area, whereas most samples of CCHC are concentrated near the ground truth. The visual impression is also confirmed by the calculation of the RMSE (see Table 5.1). The results of CCHC outranks those of RANSAC by a factor of 2.



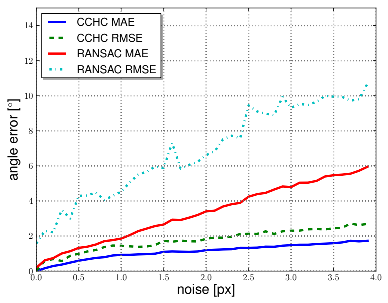
(a) 10 points in front of plane.



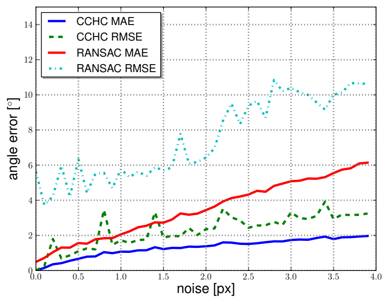
(b) 20 points in front of plane.



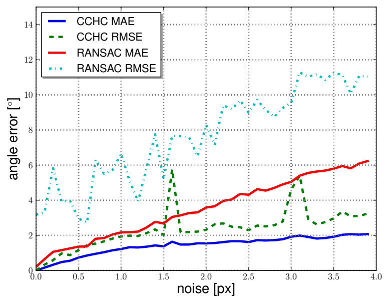
(c) 30 points in front of plane.



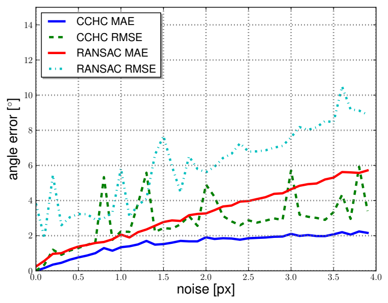
(d) 40 points in front of plane.



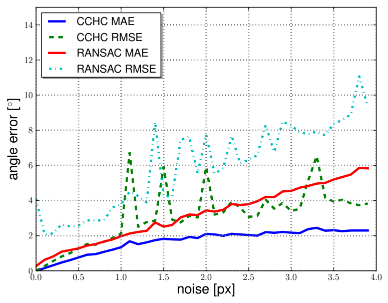
(e) 50 points in front of plane.



(f) 60 points in front of plane.

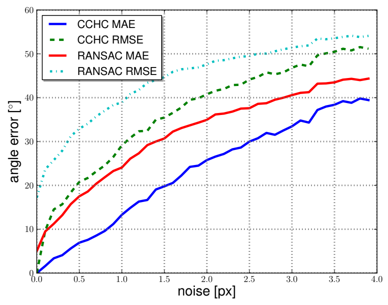


(g) 70 points in front of plane.

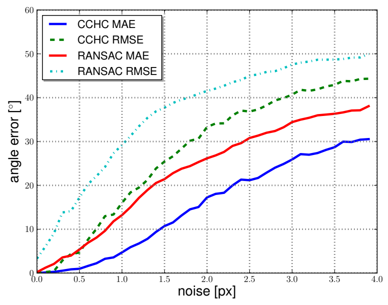


(h) 80 points in front of plane.

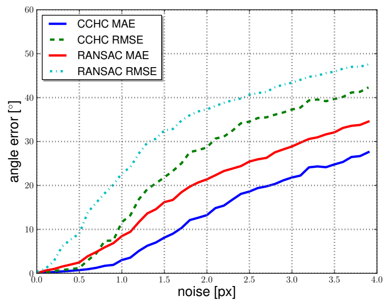
Figure 5.5: Comparison between CCHC and RANSAC (5-point algorithm).



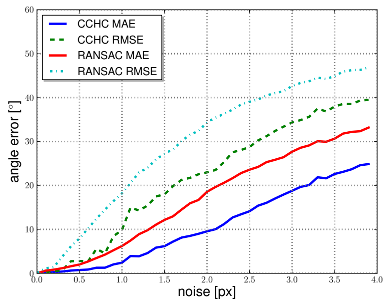
(a) 10 points in front of plane.



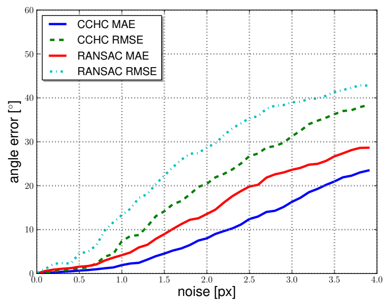
(b) 20 points in front of plane.



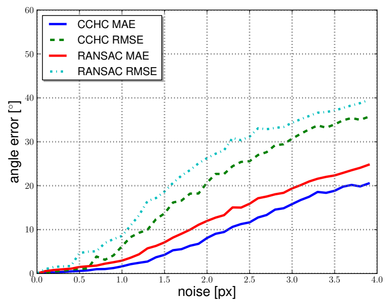
(c) 30 points in front of plane.



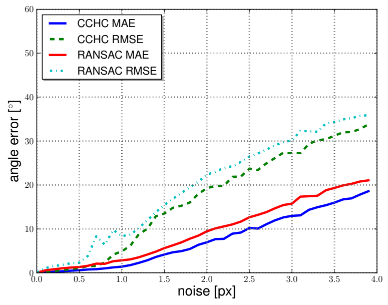
(d) 40 points in front of plane.



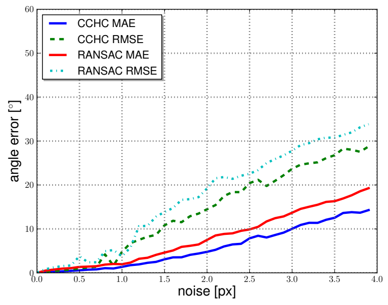
(e) 50 points in front of plane.



(f) 60 points in front of plane.

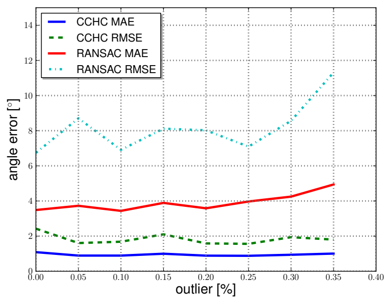


(g) 70 points in front of plane.

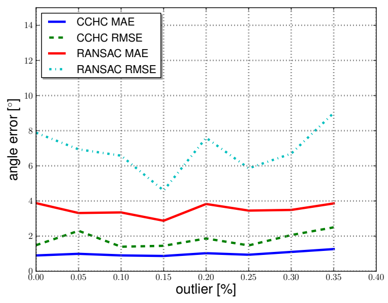


(h) 80 points in front of plane.

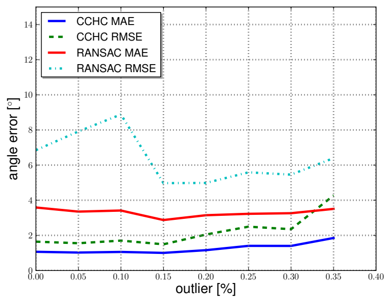
Figure 5.6: Comparison between CCHC and RANSAC (8-point algorithm).



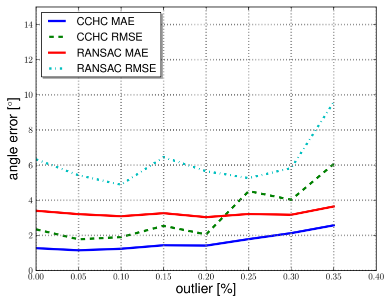
(a) 10 points in front of plane.



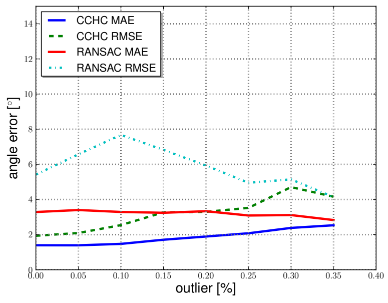
(b) 20 points in front of plane.



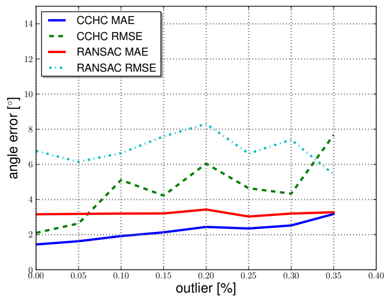
(c) 30 points in front of plane.



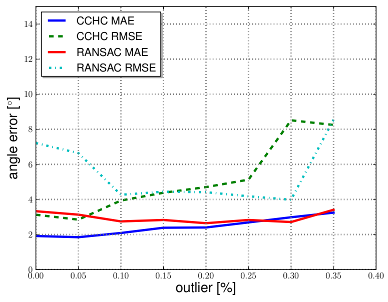
(d) 40 points in front of plane.



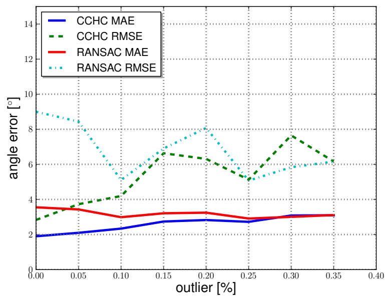
(e) 50 points in front of plane.



(f) 60 points in front of plane.

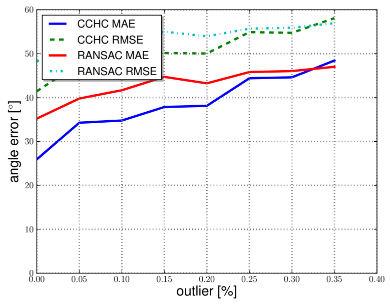


(g) 70 points in front of plane.

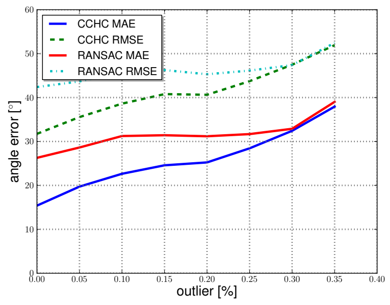


(h) 80 points in front of plane.

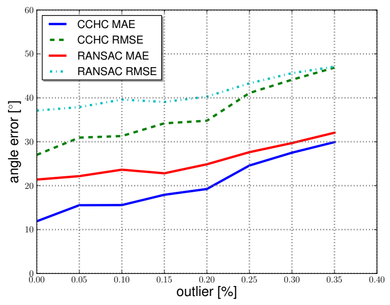
Figure 5.7: Comparison between CCHC and RANSAC (5-point algorithm) with outlier and noise=2.0.



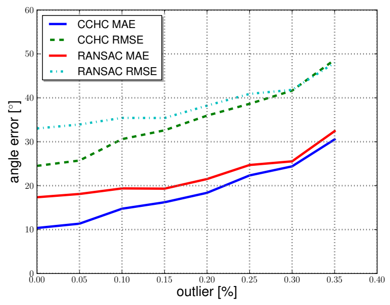
(a) 10 points in front of plane.



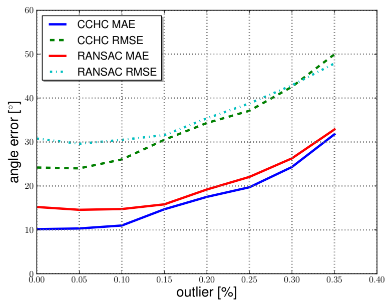
(b) 20 points in front of plane.



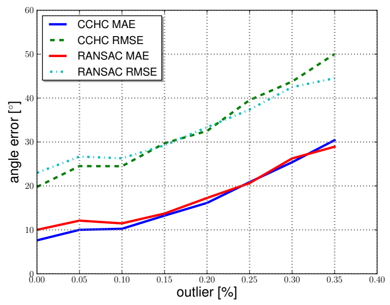
(c) 30 points in front of plane.



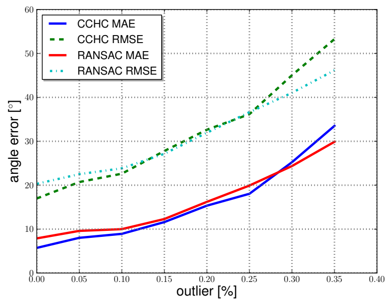
(d) 40 points in front of plane.



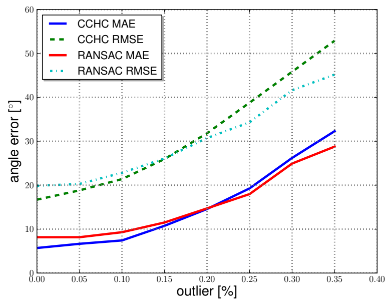
(e) 50 points in front of plane.



(f) 60 points in front of plane.



(g) 70 points in front of plane.



(h) 80 points in front of plane.

Figure 5.8: Comparison between CCHC and RANSAC (8-point algorithm) with outlier and noise=2.0.

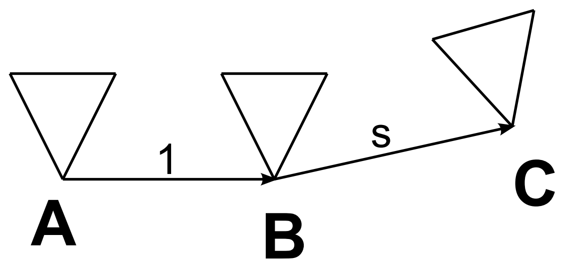


Figure 5.9: Camera triplet.

5.2 Camera Triplet

Once the relative orientation has been computed for all matched image pairs, camera triplets which have common matches can be defined. These triplets can be efficiently found by accessing the underlying data structure. For two cameras the scale is arbitrary and can only be defined by prior knowledge. The presented approach is based on camera triplets where a relative scale between camera B and C exists. It is called relative scale, as it is relative to the first camera pair (A-B) (see Figure 5.9). The complete path estimation task is reduced to the problem of iteratively concatenating triplets to generate a metric map. It is essential for concatenating triplets, that two triplets share two cameras, because a metric transformation between the overlapping cameras exists.

Each triplet is associated with a certain error. By concatenating triplets, the error increases with every triplet. It is necessary to select triplets which are considered to be most precise. The quality of a triplet depends on several parameters: Number of matches, BA error, relative scale, relative depth and distribution of matches. It can be traced back to the quality of two relative orientations and the accuracy of the relative scale. As in practice the covariance of a relative orientation is time consuming to compute, two triplets are compared using the presented parameters. A high number of matches stabilizes the configuration. The reprojection error after BA is also a valuable indicator. In the photogrammetric sense, the camera configuration is more accurate if the baseline to depth ratio his high, here denoted as the inverse, the relative depth. If both pairs (A-B,B-C) have similar relative depth, the relative scale will be close to 1.0. Additionally, the distribution of points within each image of the triplet is considered. All these attributes are concatenated in an edge cost function (Section 5.3.2).

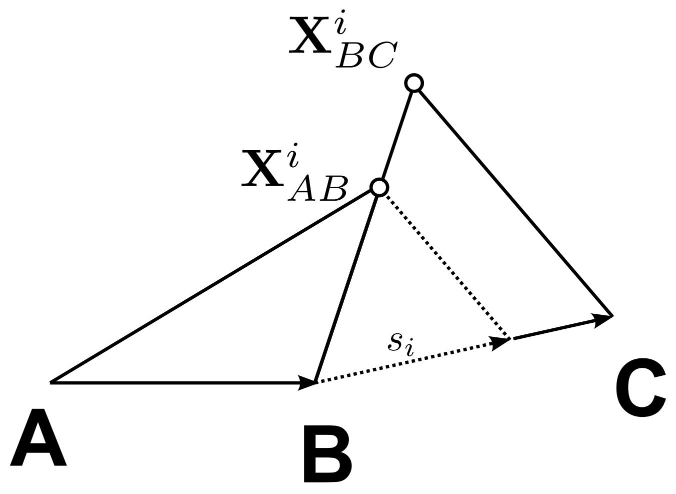


Figure 5.10: Camera triplet with scale.

5.2.1 Scale Selection

For scale selection, two relative orientations $(\mathbf{t}_{AB}, \mathbf{q}_{AB})$ and $(\mathbf{t}_{BC}, \mathbf{q}_{BC})$ are given, parametrized by the translation vector $\mathbf{t}_{AB}$ from camera A to camera B and a quaternion $\mathbf{q}_{AB}$ describing the rotation between A and B. The same applies for the tuple BC. Placing camera B in the origin as a canonical camera with the projection matrix $\mathbf{P}_B$, the projection matrix of camera A is defined as

$$\mathbf{P}_A = \mathbf{K}_A \mathbf{R}_{AB}^T [\mathbf{I}|\mathbf{t}_{AB}] \tag{5.32}$$

and

$$\mathbf{P}_C = \mathbf{K}_C \mathbf{R}_{BC} [\mathbf{I}| - s\mathbf{t}_{BC}] \tag{5.33}$$

for camera C, with $\mathbf{R}_{AB}$ and $\mathbf{R}_{BC}$ being the rotation corresponding to the quaternion $\mathbf{q}_{AB}$ and $\mathbf{q}_{BC}$, respectively. s remains the only unknown parameter. Everything else is computed as described in Section 5.1. With $s = 1.0$ the camera centers of B and C have a distance of $||\mathbf{C}_B - \mathbf{C}_C|| = 1.0$. The same holds for camera A and B. From the projection matrix $\mathbf{P}_A$ and $\mathbf{P}_B$ as well as $\mathbf{P}_B$ and $\mathbf{P}_C$, two point clouds are computed with the elements denoted as $\mathbf{X}^i_{AB}$ and $\mathbf{X}^i_{BC}$ (see Figure 5.10). To compute the scale between the three cameras, the interception theorem can be applied.

$$\frac{s_i}{||\mathbf{C}_B - \mathbf{C}_C||} = \frac{||\mathbf{C}_B - \mathbf{X}^i_{AB}||}{||\mathbf{C}_B - \mathbf{X}^i_{BC}||} \tag{5.34}$$

As the distance between camera B and C is set to 1.0, Equation 5.34 simplifies to Equation 5.35.

$$s_i = \frac{||\mathbf{C}_B - \mathbf{X}^i_{AB}||}{||\mathbf{C}_B - \mathbf{X}^i_{BC}||} \tag{5.35}$$

Each 3D point votes for one scale. For robustness the scale is selected as the quantile mean over the complete set of all computed scales.

5.2.2 Tile Variance within a Triplet

Well distributed points are essential to calculate the geometric information of a triplet. The distribution of the points can be expressed as the variance of a 2D histogram. For each image the variance of the points is computed separately. By discretizing the image in a grid of $n \times n$, the number of matched features per bin is computed. The 2D histogram $\mathbf{M}$ is normalized. All entries add up to 1.0.

$$\sigma^2 = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\mathbf{M}_{i,j} - \frac{1}{n^2})^2 \tag{5.36}$$

The final variance of a triplet is represented by the mean of three variances.

5.3 Graph Generation

The triplets, which are computed according to Section 5.2, are used to form a graph. The graph structure contains two different node types. The first type represents a triplet (yellow) with its image ids, whereas the second node type (red) represents a single image (see Figure 5.11). The graph is a directed weighted graph in which two nodes of type 1 will share an edge if two images of the triplet are identical. A connection between node type 1 and node type 2 will be established if they share the same image. The weight of each incoming edge depends on the quality of the triplet. The cost of an edge between node type 1 and 2 is 0.0. Before discussing the generation of the path, other SfM approaches which rely on a graph structure are reviewed briefly.

5.3.1 Related Approaches

Most approaches which work with a graph structure aim at handling thousands of images of online foto collections such as Flickr. They focus on the problem of finding a set of images which covers the general structure of the

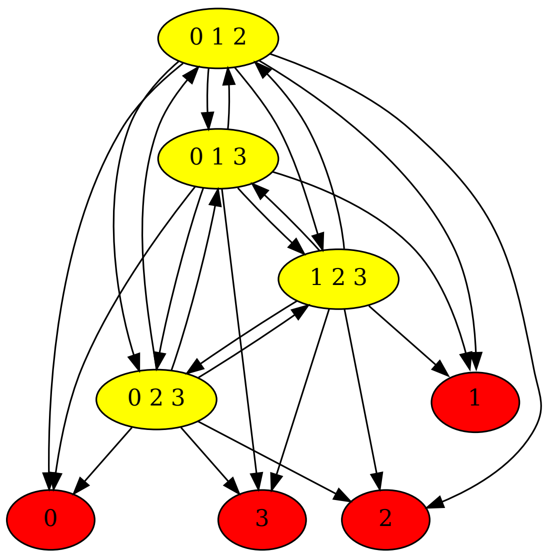


Figure 5.11: Graph representation, with yellow nodes containing triplets and red nodes representing a single camera position [94].

scene. All other images can then be included in the scene by spatial resectioning. Snavely et al. [82] generate a skeletal graph of images. Two images share an edge if the relative orientation has been successfully computed. To find the skeletal graph, they first create a maximum leaf spanning tree. This tree is transformed into a t-spanner by adding additional edges. Each edge in the graph is associated with a covariance computed in a BA step. Associating the covariance with the Hessian matrix from a non-linear optimization may not work for strongly non-linear systems. As the Hessian is the linear approximation of the current estimate, it even may not belong to the linearization of the optimal solution. In case the BA gets stuck in a local minimum, the computed covariance can mislead the algorithm. For this reason, the presented approach uses a cost function which combines measured parameters rather than using an approximated covariance whose computation is time consuming. Li et al. [43] use an even simpler method and weight the edges in the graph with the number of inliers supporting the estimated relative orientation model. Similarly, Bartelson et al. [5] construct a two view matching graph from wide baseline image sets where the number of correspondences is used as similarity value between two images. This matching graph is transformed to a maximum spanning tree. A set of triplets which meet certain criteria are derived from the graph and are used to calculate the orientation of the images. This work defines the goodness of a triplet as described in the next section.

5.3.2 Edge Weight

A graph, as described in the previous section, may contain thousands of nodes, but only a small number is considered for computing the positions of the cameras. The quality of the camera positions depends on the selected triplets. The triplets are selected according to the edge weight within the graph. A minimal cost is desirable. The definition of the edge weight has a strong influence on the camera path. Several different parameters can be taken into account to define the quality of a triplet:

- n: Number of points supporting the triplet
- ϵ: Reprojection error in pixel from the BA step
- s: Relative scale within the triplet (A,B,C)
- d_1: Relative mean depth between A and B
- d_2: Relative mean depth between B and C
- κ: Tile variance of a triplet

From these parameters an edge weight is empirically derived to select the most precise triplets. This cost function consists of five parts in which each part is related to the quality of the triplet. A small reprojection error ϵ within BA is desirable. To achieve a similar parallax between camera pair A,B and B,C, the distance between camera A and B should be close to the distance between camera B and C. This configuration is assumed to be more stable than one, where camera B and C almost overlap. A similar parallax is incorporated in the cost function as $(1-s)^2$, which holds for motions parallel to the object of interest. This formulation can also result in a small cost if the parallax is almost zero, as it will be the case if the camera moves in the viewing direction by a distance of 1.0. Since this is an unstable configuration, a second cost is included. The relative depths between camera pair A,B and B,C are compared to each other. Configurations, in which the relative depths of both pairs are similar, are preferred. It is formulated as

$$\delta = \left\{ \begin{array}{ll} (1 - \frac{d_1}{d_2})^2 & : d_1 \leq d_2 \\ (1 - \frac{d_2}{d_1})^2 & : d_2 < d_1 \end{array} \right\} . \tag{5.37}$$

Additionally, the tile variance as described in Section 5.2.2 is included in the complete cost function. Two different edge cost functions have shown to produce

Edge cost function	Mean camera difference to ground truth
$1/n$	5.59
δ	5.93
ϵ	7.88
$\|\|1-s\|\|$	8.25
κ	10.61
$(1-s)^2$	12.03
δ/n	9.27
$\epsilon\delta/n$	8.99
$\|\|1-s\|\|\epsilon\delta/n$	9.78
$\kappa\epsilon\delta/n$	**2.96**
$\kappa\|\|1-s\|\|\epsilon\delta/n$	8.56
$\kappa(1-s)^2\epsilon\delta/n$	**4.45**

Table 5.2: Evaluation of different edge cost functions.

reliable initial estimates, first

$$\frac{\kappa(1-s)^2\epsilon\delta}{n} \tag{5.38}$$

and second

$$\frac{\kappa\epsilon\delta}{n} \; . \tag{5.39}$$

Each edge is weighted by this formula. The incoming edge is the edge corresponding to the triplet, as transitioning to a new camera will always produce a cost. This cost function is evaluated empirically on the benchmark dataset Castle-P30 (Section 7.2). Several different cost functions have been evaluated and the initial path was compared to ground truth. Table 5.2 shows how good the initial path is estimated in comparison to ground truth. Many more combinations have been evaluated but only a comprehensive collection is presented here. Considering each parameter separately, Table 5.2 shows that the most important parameter is the number of matches as used in [43]. Simply combining separate parameters does not necessarily lead to a good initialisation. As the numbers are only produced on one dataset with given ground truth, the behavior may be different for other datasets. Nevertheless, reliable results have been computed with both presented cost functions.

5.4 Path Estimation

Based on the graph structure described in Section 5.3, various camera paths can be created by selecting different triplets. To compute a promising path, a stable initial node is important. The betweenness centrality is computed for each node, to select a node which is most central.

5.4.1 Betweenness Centrality

The betweenness centrality is based on the measurements of all shortest paths that exist in a graph. The general idea is that a node will have a higher centrality if it is part of many shortest paths. Assuming a set of nodes V with $u \in V$ and $t \in V$, the betweenness centrality of a node $v \in V$ is defined as

$$C_B(v) = \frac{2}{(n-1)(n-2)} \sum_{v \neq u \neq t} \frac{\sigma_{ut}(v)}{\sigma_{ut}} \, | \sigma_{ut} > 0 \tag{5.40}$$

with σ_{ut} being the number of all shortest paths from u to t and $\sigma_{ut}(v)$ being the number of all shortest path from u to t passing through v. It is assumed that at least one shortest path between u and t exists [70, 22].
Since the scope of this work lies on connecting cameras and not triplets, a betweenness centrality is computed only from paths that connect nodes of type 2. As the graph in Figure 5.11 is a directed graph with the camera nodes being sinks, the sink camera nodes are duplicated (red nodes) and inserted as source nodes (blue nodes) as shown in Figure 5.12. Now the shortest paths from each source node to all sink nodes are computed. The triplet node with the highest betweenness centrality is selected as starting node.
This algorithm has a complexity of $\mathcal{O}(n^2)$ in the number of cameras when the number of triplets is not considered. The number of triplets influences the complexity of finding the shortest path. The computation of the shortest paths can be time consuming for many cameras. An approximate solution can be found by generating groups of cameras. Since the images are supposed to be taken as a sequence, they follow a spatial order. A set of consecutive images, which will be close in space, can be grouped together. In the graph representation extra group nodes are included, which are either connected to source or to sink camera nodes (see Figure 5.13). The approximate solution is established by computing the betweenness centrality between the groups instead of single cameras.

5.4.2 Camera Pose

Once the initial node is selected, the pose of each image is computed by propagating the pose over a list of image triplets. The triplets are selected by

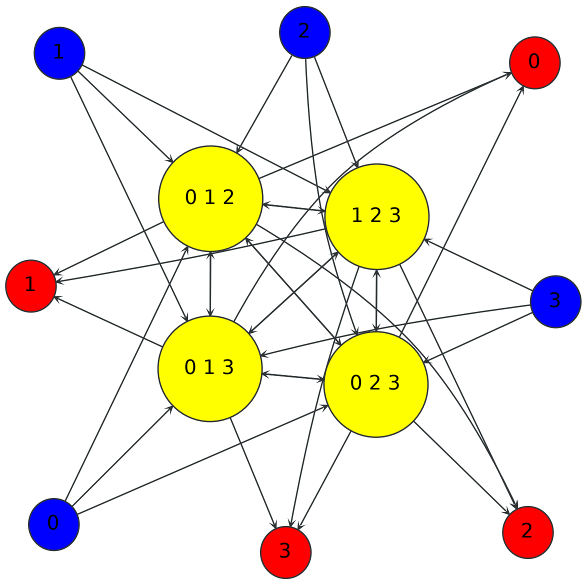


Figure 5.12: Graph with additional sink camera nodes in blue.

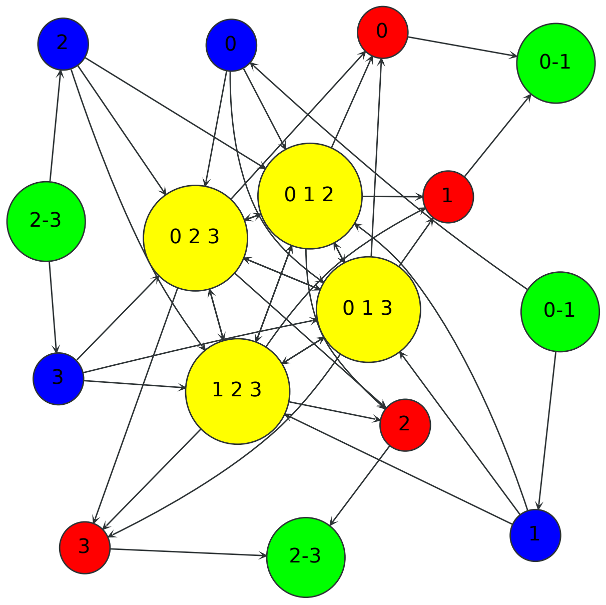


Figure 5.13: Graph with additional group nodes in green.

computing the shortest path to each image from the most central node. Each triplet defines a local coordinate system between three images which differs only in one similarity transformation to the global coordinate system. Since two triplets which are connected within the graph always share two identical images the overall scale can be propagated from one triplet to the other. For the initial node the first image A is set to the origin looking in z-direction. The second image B is placed at $\mathbf{C}_B = \mathbf{t}_{AB}$ with a rotation $\mathbf{q}_B = \mathbf{q}_{AB}$. $||\mathbf{t}_{AB}||$ defines the scale of the global coordinate system. The camera parameters of the third image C is defined as

$$\tilde{\mathbf{C}}_C = s_{AB} s_{ABC} \mathbf{q}_B^{-1} \tilde{\mathbf{t}}_{BC} \mathbf{q}_B + \tilde{\mathbf{C}}_B \tag{5.41}$$

$$\mathbf{q}_C = \mathbf{q}_{BC} \mathbf{q}_B \tag{5.42}$$

with s_{AB} being the global scale between images A and B and s_{ABC} being the relative scale of the triplet between image A, B, and C. Following this procedure, the next triplet is selected. Two images of the new triplet are already reconstructed with a known position. These two images are inserted into Equations 5.41 and 5.42 as image A and B to compute the parameters of the third image and insert it into the global coordinate system.

5.4.3 Camera Pose Optimization

In all SfM approaches, BA is the most important part. The non-linear least squares optimization is added as a final refinement step. Snavely et al. [81] used BA in each step an image is added to the path. Applying BA in each step results in a high computational cost. However, it stabilizes the result. Bundler has become a popular piece of software which is used in many different areas for reconstructing 3D scenes, e.g. archeology and geodesy.
BA can be time consuming especially when dealing with large scenes. Agarwal et al. [1] investigated the scaling of different BA methods for large scenes. Recently different approaches have been presented to accelerate BA [11, 32, 98].
The next section reviews the classical BA method before presenting a novel optimization scheme inspired by the parametrization of [12] originally developed for extended Kalman filter (EKF). Further, a general optimization framework, OpenOF, is presented which is successfully used for arbitrary least squares optimizations [95]. The framework was developed during this work and made publically available under GNU GPL v3 license. OpenOF has the advantage over other BA implementations e.g. Sparse Bundle Adjustment (SBA) [49], Simple

Sparse Bundle Adjustment (SSBA) [100] and Parallel Bundle Adjustment (PBA) [98] that not only the reprojection error but rather arbitrary cost functions can be minimized. All known implementations have some kind of limitations. Most often, the open source implementations assume that all images are either taken by the same camera (all internal parameters are identical) or all images are from arbitrary cameras [98, 100]. OpenOF can also handle special setups, such as several cameras are mounted on a rig.

5.4.3.1 Bundle Adjustment

The name bundle adjustment refers to the rays of light beginning at 3D points and projecting into different cameras. This bundle of rays between 3D points and images is optimized with a non-linear optimization algorithm as described in Section 2.4.1. The parameters which are optimized belong to a set of n 3D points $\{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{n-1}, \mathbf{X}_n\}$ and m images $\{C_1, C_2, \ldots, C_{m-1}, C_m\}$. Additionally, measurements $\mathbf{x}_{ij}$ represent the projection of a point i in image j. The function that is optimized is defined in Equation 5.43, where f is the projection of a 3D point to image coordinates, defined in Equation 2.3.

$$\underset{\forall \mathbf{X}, \forall C}{\operatorname{argmin}} \sum_{\forall \mathbf{x}_{ij}} (\mathbf{x}_{ij} - f(\mathbf{X}_i, C_j))^2 \tag{5.43}$$

Usually, one 3D point is only present in a limited number of images. This results in a sparse structure of $\mathbf{J}^T\mathbf{J}$ (see Equation 2.23). Modelling the sparse structure, as described in Section 2.4.1, avoids tremendous computation time, as even a middle-sized problem consists of thousands of 3D points and hundreds of images. Popular approaches, such as SBA [49], utilize the Schur complement for efficiently solving the normal equation within the Levenberg-Marquardt algorithm. This works well due to the special sparse structure of standard BA. Yet, with increasing complexity of the structure the Schur complement has its limitations. If the images are all taken with the same camera, the internal calibration is constrained to be the same for all images. In OpenOF it is also possible to define groups of images which have the same internal calibration. Further constraints can be easily integrated e.g. several cameras mounted on a car having a fixed distance and orientation to each other.

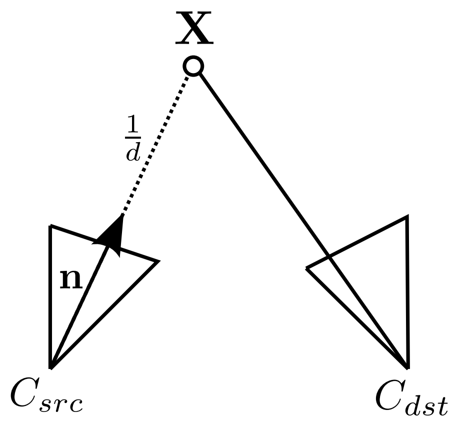


Figure 5.14: Inverse point parametrization with source and destination camera [95].

5.4.3.2 Inverse Bundle Adjustment

The inverse BA [95] is a ray based parametrization. In contrast to BA, a 3D point is not free in space, but it is attached to the image in which it first appeared in the processing chain. A 3D point is parametrized by a source camera C_{src}, a unit vector $\mathbf{n}$ and the inverse distance d (see Figure 5.14). More formally speaking, the 3D point $\mathbf{X}$ is defined in quaternion notation as

$$\tilde{\mathbf{X}} = \frac{1}{d}\mathbf{q}^{-1}\tilde{\mathbf{n}}\mathbf{q} + \tilde{\mathbf{C}} \tag{5.44}$$

with $\mathbf{q}$ being the quaternion representing the rotation and $\mathbf{C}$ denoting the camera center of the source camera. The optimization function changes from Equation 5.43 to Equation 5.45. The derivative of this function is more complex than before since $\mathbf{X}$ is a function itself.

$$\underset{\forall \mathbf{X}, \forall C}{\operatorname{argmin}} \sum_{\forall \mathbf{x}_{ij}} (\mathbf{x}_{ij} - f(\mathbf{X}_i(C_{src(i)}, \mathbf{n}, d), C_j))^2 \tag{5.45}$$

The big advantage of this approach is the reduction of parameters by $2n$. Additionally, the number of measurements is reduced because no projection for the source camera is necessary. The ratio between parameters and measurements is most important for the convergence. Assuming a fixed intrinsic calibration, the ratio of the inverse parametrization is smaller than the ratio of the standard parametrization, for $n > 7$, $m > 3$.

$$\frac{7m + n}{2n(m-1)} < \frac{7m + 3n}{2nm} \tag{5.46}$$

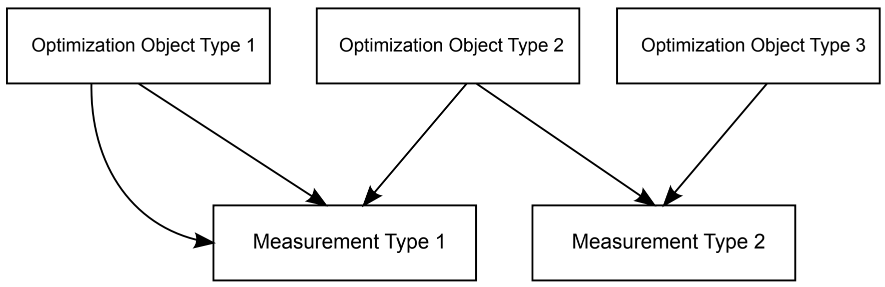

Optimization Object Type 1
Optimization Object Type 2
Optimization Object Type 3
Measurement Type 1
Measurement Type 2

Figure 5.15: Optimization objects interact with different measurement types.

The reduction of the parameter space also leads to a disadvantage. A 3D point cannot acquire arbitrary positions in space. Since the point is fixed to the viewing ray of the source camera, the only free parameter is the depth. The image orientation has to change, to take in an arbitrary position. Since all points enforce a certain orientation of the image, arbitrary 3D point positions are not possible.

The adjusted parameters, which are more likely to be closer to the optimal solution than the initial parameters, can be further optimized with a standard BA parametrization to overcome this disadvantage.

5.4.3.3 OpenOF

OpenOF [95] is a general framework for least squares optimization which was developed within this work. It is build in order to efficiently combine general cost functions. The sparse structure of BA is incorporated by using a coordinate list (COO) matrix format. Each value is stored by its row and column. To solve the non-linear least squares system, OpenOF follows the algorithms presented in Section 2.4. With sparseLM [48] and g2o [41] two alternative libraries are available to solve this general problem. Due to the modular structure, OpenOF can incorporate any kind of robust cost function. Similar to g2o, where only the Huber cost function can be used, OpenOF can define for any measurement a different robust cost function. Currently, the robust cost functions presented in Section 2.4.3 are integrated.

In contrast to other approaches, OpenOF allows for an easy integration of new models. The optimization scheme follows the layout shown in Figure 5.15. The parameters to be optimized are grouped into so called optimization objects. In case of BA, there are four optimization objects: 3D point, 2D measurement, internal camera parameters, external camera parameters. Two types of parameters exist for each object: On the one hand parameters that can be modified by the optimization and on the other hand constant parameters. In case of sensor data, all parameters belong to the second type and are constant. For some models it might be useful to include the parameters as variable and constant e.g. a position of a vehicle is given by different onboard sensors. The position is further adjusted respecting the variance of the different sensors.

These optimization objects serve as input to measurement types which represent a cost function. Each measurement type defines one cost function, e.g. the reprojection error. To have a wide diversity in the design of an optimization, it is possible to hold an optimization object constant in one measurement type and refine it in a different measurement type. While in other approaches the Jacobi matrix of the cost function needs to be provided, OpenOF generates a sparse Jacobi matrix by symbolic differentiation. The objective function is defined in the high level scripting language python. The python module sympy [88], which is an open source library for symbolic mathematics, makes it possible to define complex cost functions in an abstract and readable manner. High level libraries often come at the expense of speed. Since least squares optimization is highly demanding regarding computational cost, this problem is overcome by fast c++ code which runs on NVIDIA graphics cards and is automatically generated from the scripting language. OpenOF is successfully used in ongoing projects for refining drift and position parameters of satellites [29] as well as a project for dense 3D reconstruction [92]. In the latter case, algebraic as well as numerical differentiations are mixed.

5.4.3.4 Evaluation

For evaluation, inverse BA is compared against BA, both computed with OpenOF. Both methods are evaluated against simulated data as well as a real world dataset. The performance is compared with two open source BA implementations, SSBA [100] and PBA [98]. The latter is implemented for the GPU and CPU, respectively. In contrast to SSBA, PBA uses CG similar to OpenOF. The performance of OpenOF is shown to be comparable to the specially optimized PBA. All evaluations are performed on an Intel i7 with 3.07 GHz, 24 GB RAM and a NVIDIA GTX 570 graphics card with 2.5 GB memory. Synthetic data shown in Figure 5.16 is used for evaluation. The 3D points are randomly generated and placed on the walls of a cube. The points are only projected into cameras for which the points are directly visible in reality,

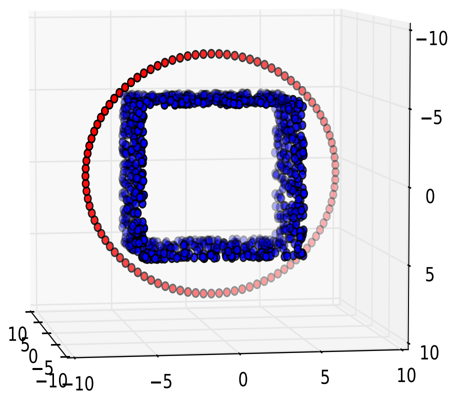
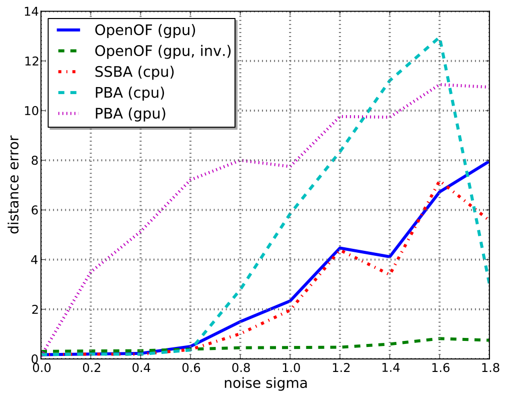
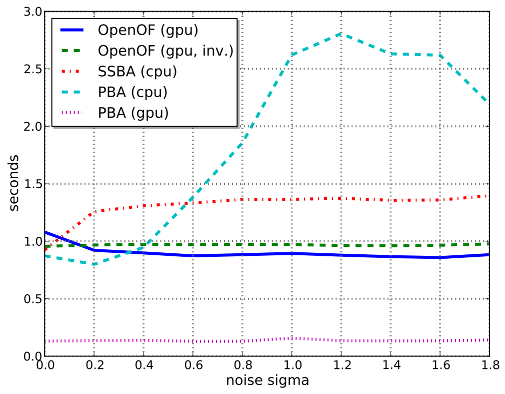


Figure 5.16: Setup of points (blue) and cameras (red) for the simulated data [95].

assuming the cube is not transparent. The cube has a size of $10 \times 10 \times 10$. The cameras are positioned on a circle with a radius of 8, facing the center of the cube. The points are projected into a virtual camera with an image size of 640×480 and a focal length of 1000 measured in pixel. The principal point is defined to lie in the center of the image. To simulate SfM approaches where the initial camera position is rather a rough estimate, Gaussian noise is added to the camera position. Normally distributed noise of 0.5 pixels is applied to the measurements. Furthermore, new points are triangulated from the noisy projections instead of using the original 3D point, resulting in a more realistic setup. With known positions of the cameras, the convergence of the five BA implementations introduced above is investigated by continuously increasing the noise of the camera position. Within the simulation 100 cameras and 1000 points are used. The overall estimate of each method is transformed to the original cameras with a Helmert transformation. The distance error between the estimated and true camera position is shown in Figure 5.17(a) for different amounts of noise. For each noise level on the camera position the mean of 5 iterations is plotted. Within our test the GPU implementation of PBA did not converge in most cases, as shown in Figure 5.17(a). Every other method shows a similar precision for $\sigma < 0.6$. For $\sigma > 0.6$, inverse BA shows a better convergence than the other methods due to fewer degrees of freedom. Especially the CPU version of PBA most rarely converged for noise $\sigma > 1.2$. The convergence problem of PBA results in increased computation time (Figure 5.17(b)), while the other approaches stay constant in time. To evaluate the speed of the algorithms, each method runs with a constant amount of Gaussian noise while concomitantly increasing the number of cameras. This reduces the distance

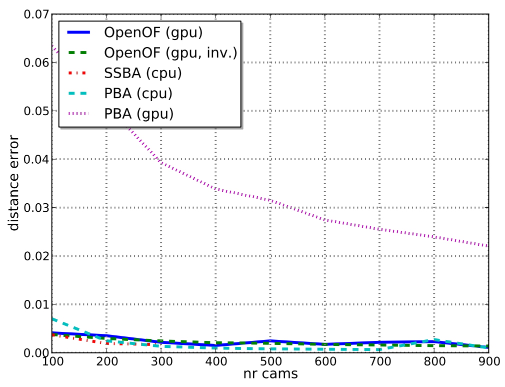


(a) Precision

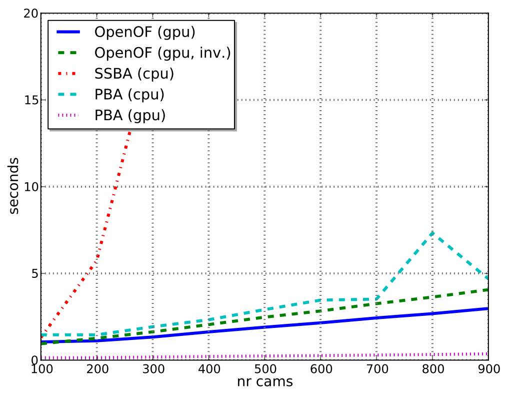


(b) Performance

Figure 5.17: Applying normally distributed noise on the camera positions (100 cameras) [95].



(a) Precision

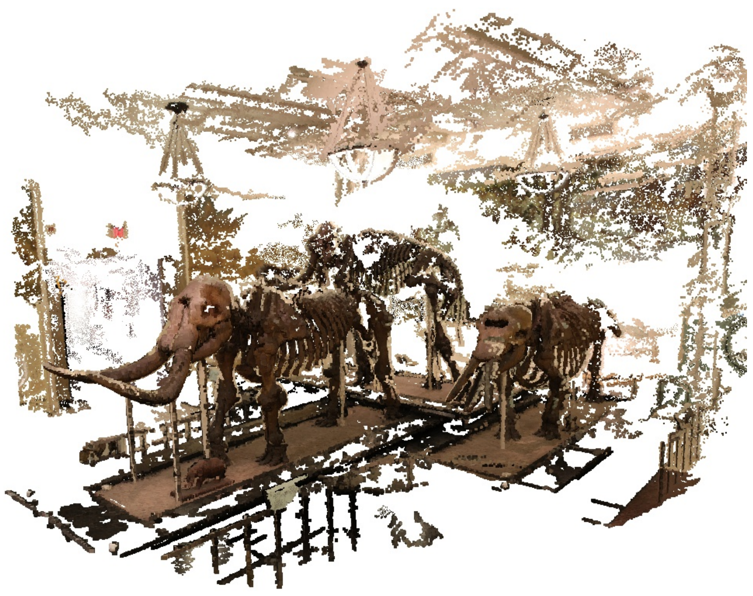


(b) Performance

Figure 5.18: Changing the number of cameras with constant amount of noise on the camera position ($\sigma = 0.5$) [95].

between two cameras and increases the number of projections of one 3D point. As expected, SSBA, using a direct solver implemented on a CPU, shows the lowest performance. Regarding the described example, the approaches using CG (OpenOF and PBA) operate much faster than the one with a direct solver (see Figures 5.18(a) and 5.18(b)). The GPU version of PBA cannot be taken into account because it did not converge.

The real world data evaluation is performed on a dataset with 390 images taken with a Canon EOS 7D camera (see Figure 5.19(b)). The pictures were taken in the American Museum of Natural History in New York within the scope of a project aiming at the reconstruction of extinct animals. Figure 5.19(a) shows the final reconstruction computed with PMVS2 [24]. The path computed with


(a) Quasi-dense point cloud

(b) Sample image

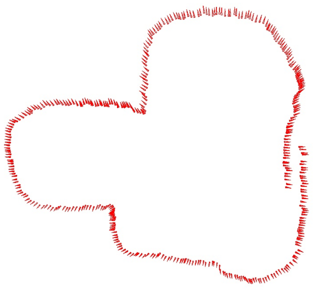
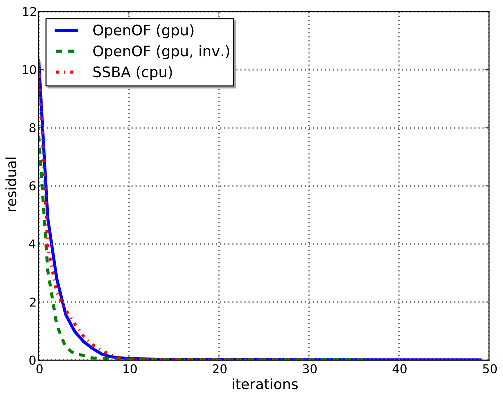
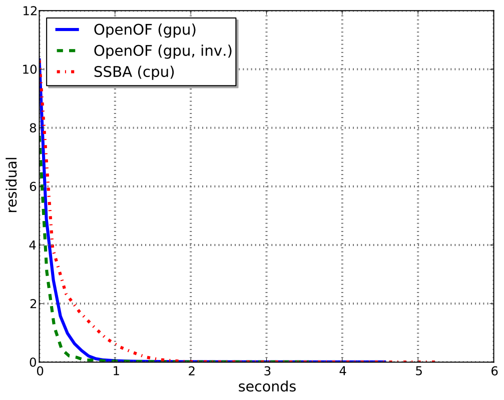

(c) Camera path

Figure 5.19: Three mammoths acquired at the American Museum of Natural History [95].



(a) Iterations



(b) Time

Figure 5.20: Convergence of the mammoth dataset [95].

OpenOF is illustrated in Figure 5.19(c). The convergence rates of SSBA and OpenOF are compared against each other. PBA is not included in this part of the evaluation due to a different metric system for the residuals which could not be adapted. As shown in Figure 5.20(a), the convergence of SSBA and OpenOF is similar for BA with respect to the number of iterations. Regarding the number of iterations relative to the overall time, the CG approach outperforms the direct solver. This speedup is even larger for datasets where the amount of overlapping images is higher, as shown in the synthetic evaluation. The convergence rate of inverse BA is better than of BA.

5.4.4 Merging Reconstructions

A single object of interest is frequently acquired multiple times. There are different reasons for capturing multiple datasets of the same object: Either parts of the objects which were missed in the first round are captured in a second iteration or the images are acquired with different distances to the object. The precision of a reconstruction can be computed by merging multiple reconstructions and estimating discrepancy between those.
It is assumed that each project contains a list of 3D points and their 2D correspondences (see Figure 3.8 for a complete overview of the given data structure.). From two sets of 3D points the subset of points which is identical in both projects is identified and the correspondences between those points is established. The SIFT descriptor is used to establish the correspondences. Since each 3D point is constructed from multiple 2D points, a 3D point can be described by multiple SIFT descriptors. For simplicity only one descriptor is chosen for each point. Li et al. [44] proposed to either take the average of the descriptors or to take all descriptors. In this work a 3D point is represented by the descriptor which is closest to all other descriptors. The task of merging two reconstructions is reduced to matching two sets of descriptors as described in Section 4.1. Once the correspondences are established, the transformation between the two projects is computed with a Helmert transformation. Since the correspondences may contain outliers, the Helmert transformation is computed with a robust L2 truncated function. This method is repeated until all reconstructions are merged.

CHAPTER VI

Loop Closure

When acquiring an image sequence, it is common to move in a circle around the object and return to the starting point. Identifying such a situation is known in literature as 'loop closure detection' [4]. Revisiting a location can be identified with two different approaches, whereby both approaches have advantages and disadvantages. Combining both approaches can help to achieve a stable and reliable solution. The first approach is based on the visual appearance of the scene while the second makes use of the pose of the camera (spatial loop closure). Identifying the loop closure only by visual appearance may cause errors, especially if repetitive structures are present in the scene. Due to symmetric structures, opposite walls of a room may look similar. Humans are able to differentiate between two scenes by identifying small objects that are missing in the other image (see Figure 6.1) but many examples exist where even those small differences are missing. In such a case only spatial loop closure detection can be applied. The drawbacks of spatial loop closure detection become visible in long image sequences such as the car sequence presented in Section 7.4. The position error increases with every image. At the end the error may have become too large to detect a spatial loop closure. Combining both approaches leads to a robust solution.

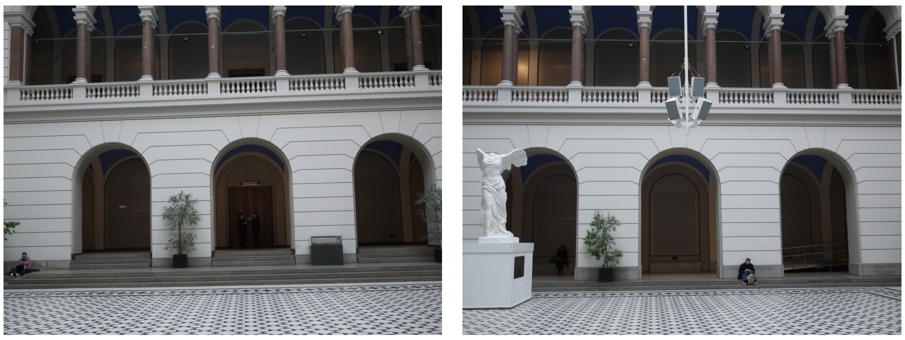


Figure 6.1: Example scene which might mislead an appearance based loop closure detection.

6.1 Image Based Loop Closure Detection

Humans are able to recognize previously visited places by their visual appearance. For loop closure detection, the content of the image plays an important role as well. The most straight-forward approach is a linear search in all previously captured images: A new image is matched with all previous images. If the number of matches exceeds a certain threshold, it will be most likely that a loop closure is detected. Since image matching is time consuming, the amount of image pairs that are matched needs to be limited. The image based loop closure detection degrades to the problem of finding images which are likely to match. It does not necessarily mean that a loop closure exists, if two images match.

A common approach defines for each image a bag of visual words [66] by inserting the descriptors in a vocabulary tree. Each leaf node of the tree represents a visual word. In this thesis, a new approach is proposed which describes one image only by one descriptor. All features which describe an image are merged by feature pooling [9]. In contrast to other feature pooling methods, the variance which appears in each dimension of the SIFT descriptors is used to form the new descriptor. Two images are assigned as matching candidate if the cosine similarity between the descriptors is below a certain threshold. In the next section vocabulary trees are described in detail, followed by a novel more effective method, the VDA.

6.1.1 Vocabulary Trees

Vocabulary trees are by now the most common way to detect loop closures using only images. The basic idea was first presented in [80]. Nevertheless, it was not spread until [66] in the SLAM community. Vocabulary trees rely on so called ‘visual words‘. Based on millions of SIFT descriptors of different objects a hierarchical k-means clustering is constructed in feature space. Each cluster center represents a visual word. From the hierarchical clustering a vocabulary tree is generated. The leaf nodes are identical with the visual words. Generating the clusters is time consuming. However, the resulting tree is not only discriminative for specific training data, but can produce reliable results for rather general images.
Two sets of images exist: The database and the query images. All features of the database images are inserted in the vocabulary tree. Each leaf node consists of a

list containing the image indices of the inserted features (inverted files).
For each visual word the Term Frequency Inverse Document Frequency (TF-IDF) is applied, originally invented for text mining, to calculate the relevance of each visual word regarding an image. The TF-IDF consists of two parts which are multiplied with each other. The first part is the Term Frequency (TF) defined as the quotient of the frequency of a visual word $n_{i,w}$ in an image i and the total number of visual words n_i in the image. The complete TF-IDF is defined as

$$tfidf(i,w) = \frac{n_{i,w}}{n_i} log \frac{N}{N_w} \tag{6.1}$$

where N is the total number of images and N_w is the total number of images where the word w appears. A sparse descriptor with a length equal to the number of leaf nodes of the tree is created for each image. This descriptor contains the scores of the TF-IDF for the visual words occurring in this image. The descriptor is normalized afterwards.

For querying an image, the normalized difference between the query descriptor and the database descriptors can be efficiently computed. The best n matches from the database are selected as loop closure candidates.

6.1.2 Variance Descriptor Analysis

VDA is a new method proposed in this thesis which describes an image with a single vector and computes an appearance based similarity value between images. This descriptor has only a size 128 and can be computed in a complete parallel manner. Detecting loop closures with vocabulary trees is a reliable, well established method, but is computationally costly. For devices with less computational power such as mobile robots or smart phones efficient methods like VDA are important.
Similar to vocabulary trees, VDA produces a set of matching candidates which is only a small percentage of all possible candidates. Considering m images, the aim is to build a similarity matrix $\mathbf{M}$ containing the results of a cost function $sim(i,j)$ for image i and image j. To compute $sim(i,j)$, a new descriptor, called Variance Descriptor (VD), is created for each image. Since two images will match if many of the SIFT features of one image have a correspondence in the second image, a new descriptor is computed out of all features of one image. The challenge is to achieve an efficient data reduction while remaining discriminative. As shown by TBH, data can be distinctively divided in a 128 dimensional space according to

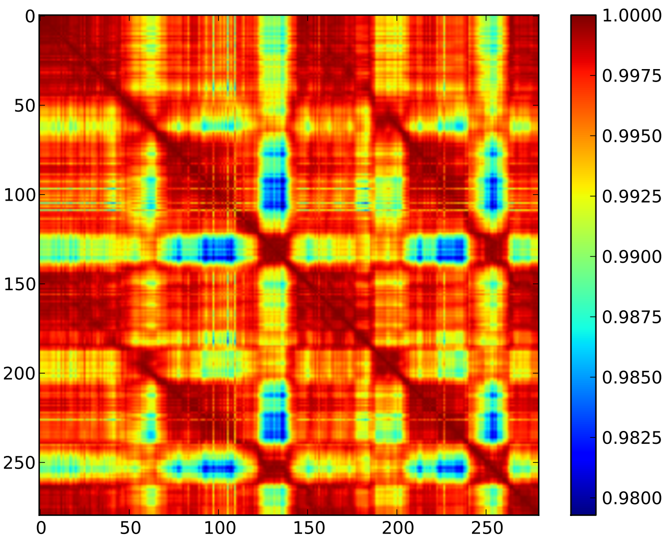


Figure 6.2: Appearance similarity matrix (VDA) of KPM dataset.

its variance. Thus, the new descriptor contains the variance over each dimension. More formally speaking, given a descriptor $\mathbf{D}_f^i$ of a feature f from image i, the VD $\mathbf{V}^i$ is computed as in [94] with n being the number of features of image i.

$$\mathbf{V}^i = \frac{1}{n-1}\sum_{f=1}^{n}\left(\mathbf{D}_f^i - \overline{\mathbf{D}}^i\right)^2 \tag{6.2}$$

The similarity $sim(i,j)$ is computed as the cosine similarity, the inner product of the normalized vectors.

$$\mathbf{M}_{i,j} = sim(i,j) = \frac{\mathbf{V}^{iT}\mathbf{V}^j}{||\mathbf{V}^i||||\mathbf{V}^j||} \tag{6.3}$$

The angle between $\mathbf{V}^i$ and $\mathbf{V}^j$ remains smaller than 90 degree, because each value in $\mathbf{V}$ is positive as the variance is always positive, resulting in a similarity value in the range 0.0 to 1.0. Figure 6.2 shows a similarity matrix for a path surrounding an object twice. The path belongs to the Königliche Porzellan Manufaktur (KPM) dataset in Section 7.3.

6.1.3 Evaluation

The following section compares the results achieved by VDA with those generated by the use of vocabulary trees. The chosen dataset belongs to the court yard of the KPM. The court yard was circled two times. The ground truth for a binary similarity matrix is generated by matching each image with all other

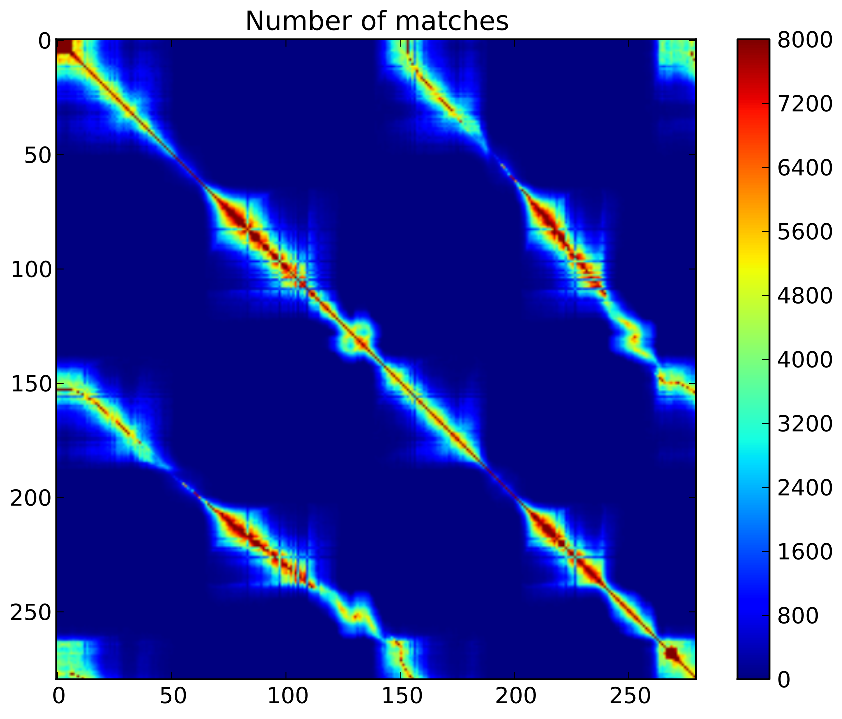
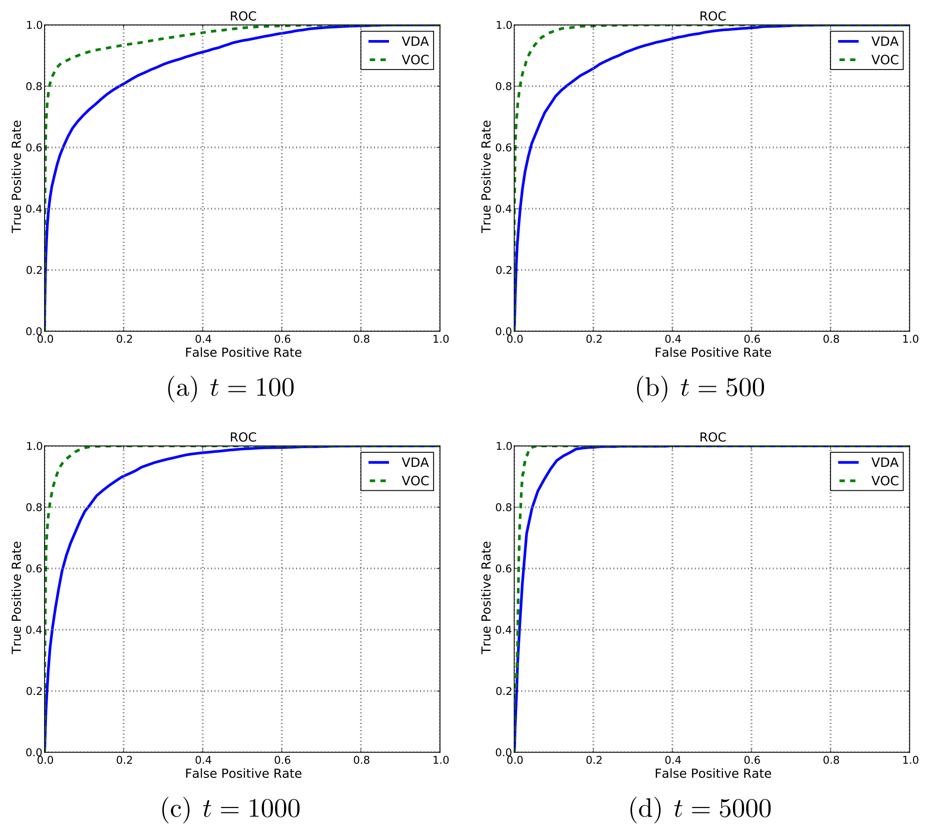


Figure 6.3: Number of matches of the KPM dataset.

images. In Figure 6.3, the maximal number of matches is set to 8000 for better visual appearance. VDA and vocabulary trees are evaluated with a Receiver Operating Characteristic (ROC) curve in Figure 6.4 considering different number of matches as ground truth. A true positive count will be achieved if a similarity is detected and the ground truth in Figure 6.3 has more than t matches. A false positive event will be counted if a similarity is detected, but the two images have less than t corresponding matches. Applying a threshold to the similarity matrix in Figure 6.2 transforms the score into a binary decision. By changing the threshold, the ROC curve is computed. The chosen dataset does not contain any misleading symmetric structures. Otherwise, the automatic ground truth generation, which is used in this example, would not be valid. Further evaluations on a dataset with symmetric structures are presented in Section 7.1.1 on the Faller Train Station (FTS) dataset. The results in Figure 6.4 show a remarkably high recognition rate for vocabulary trees with a true positive rate of 95 % with less than 10 % false positives for $t > 500$. Nevertheless, VDA achieves also a high recognition rate considering the substantial data reduction, which was applied. A vocabulary tree with 1 million visual words were used, resulting in a descriptor size of 1 million. In contrast, the descriptor size of VDA is only 128. VDA is superior to vocabulary trees concerning resources. On the FTS dataset in



(a) $t = 100$ (b) $t = 500$

(c) $t = 1000$ (d) $t = 5000$

Figure 6.4: ROC for VDA and vocabulary trees (VOC).

Section 7.1 a speedup of a factor of 350 was achieved. As a conclusion, VDA should be applied if speed and limited resources have to be considered. If speed is of second rank, vocabulary trees will usually yield a better recognition rate.

6.2 Spatial Loop Closure Detection

After successfully creating an initial path, a spatial loop closure detection can be applied to refine the results. Usually, the user returns to a position near the point where he started taking the images of the object. However, to discriminate this common scenario from exceptions, a fast and reliable approach is essential. The algorithm presented here uses a set of camera positions and a sparse point cloud. Usually, one 3D point is captured by several cameras. At first, the mean viewing direction $\mathbf{n}_i$ of each point to all corresponding cameras which triangulated this point is computed. This approximates the direction to the images corresponding to the point. A similar direction of a query image might lead to a loop closure detection.

$$\mathbf{n}_i = \frac{1}{m}\sum_{j=1}^{m}\frac{\hat{\mathbf{C}}_j - \hat{\mathbf{X}}_i}{||\hat{\mathbf{C}}_j - \hat{\mathbf{X}}_i||} \tag{6.4}$$

A list of cameras, which were part in the triangulation process, is associated with the respective 3D point. For each point the angle between the viewing ray computed in Equation 6.4 and the ray which goes from the camera center to the point is calculated and a threshold α is applied

$$\arccos\left(\frac{\mathbf{n}_i^T(\hat{\mathbf{C}} - \hat{\mathbf{X}})}{||\mathbf{n}_i||||(\hat{\mathbf{C}} - \hat{\mathbf{X}})||}\right) < \alpha \tag{6.5}$$

with $\mathbf{X}$ being the location of the 3D point and $\mathbf{C}$ being the camera center. In cases where the angle is smaller than the threshold α the image lies in the same direction as the images which triangulated the point. In this case the point votes for a loop closure regarding the respective image. Else, a negative vote is applied. Equation 6.5 is applied for all points and all cameras leading to a complexity of $\mathcal{O}(nm)$. If the angle difference is larger than α the camera may still be able to see the point, but the feature matcher will not be able to match the corresponding features within the images as the viewpoint difference is too large. All votes are stored in a similarity matrix. A loop closure is detected for each positive entry in the similarity matrix. This voting scheme has the advantage that no fixed threshold on the similarity matrix needs to be applied. A loop closure candidate is accepted if half of the sparse points correspond to an image vote for a loop closure, meaning a positive entry in the similarity matrix. The algorithm is outlined in Algorithm 3 with n points and m images.

Evaluation

In the KPM dataset loops need to be closed. The initially estimated path, shown in Figure 6.5, produced duplicated walls due to accumulated errors. Only after matching the images forming a loop, a correct result was achieved (see Figure 6.6). Since the camera positions of the second circle around the courtyard is still in a short range to the correct position, the presented spatial loop closure procedure works well. The accumulated votes as well as a binary image of the spatial similarity matrix are shown in Figure 6.7. This binary image is used to match image pairs that have not been matched before. From the new matches, potential

Algorithm 3 Generating spatial similarity matrix

```
for $i = 1$ to $n$ do
  for $j = 1$ to $m$ do
    if $\angle(\mathbf{C}_j - \mathbf{X}_i, \mathbf{n}_i) < \alpha$ then
      for all id in getIds($\mathbf{X}_i$) do
        $\mathbf{M}_{j,id} = \mathbf{M}_{j,id} + 1$
        $\mathbf{M}_{id,j} = \mathbf{M}_{id,j} + 1$
      end for
    else
      for all id in getIds($\mathbf{X}_i$) do
        $\mathbf{M}_{j,id} = \mathbf{M}_{j,id} - 1$
        $\mathbf{M}_{id,j} = \mathbf{M}_{id,j} - 1$
      end for
    end if
  end for
end for
```

new triplets arise. These triplets are computed and the respective matches are marked as filtered if they pass the filter step described in Section 4.1.4.3. Once the global ids are regenerated, the path is computed once again and BA is able to close the loop with the new correspondences. The resulting sparse point cloud is shown in Figure 6.6. The duplicated walls have vanished and the images of the two circles are close together, as the driver took the same path for the long straights.

6.3 Closing the Loop

In case of a long camera path with more than 1000 images, the gap between the cameras forming the loop can be large. If only the point correspondences are given, BA might not be able to close the loop. Especially when using a robust optimization method, these measurements are handled as outliers. In such a case it proved to be useful to distribute the errors equally between the cameras which are in the triplet propagation chain between the cameras which form the loop. Considering a begin image with a frame number f_B and an end image with the number f_E, all images for which $f < f_B$ hold are supposed to be in the propagation chain situated before the begin of the loop. Images for which $f > f_E$ holds are after the loop. All other images are in consecutive order by their frame id within the loop. In the following formulas it is assumed that cameras f_B and f_E are identical in position and rotation. This is valid if the image is included

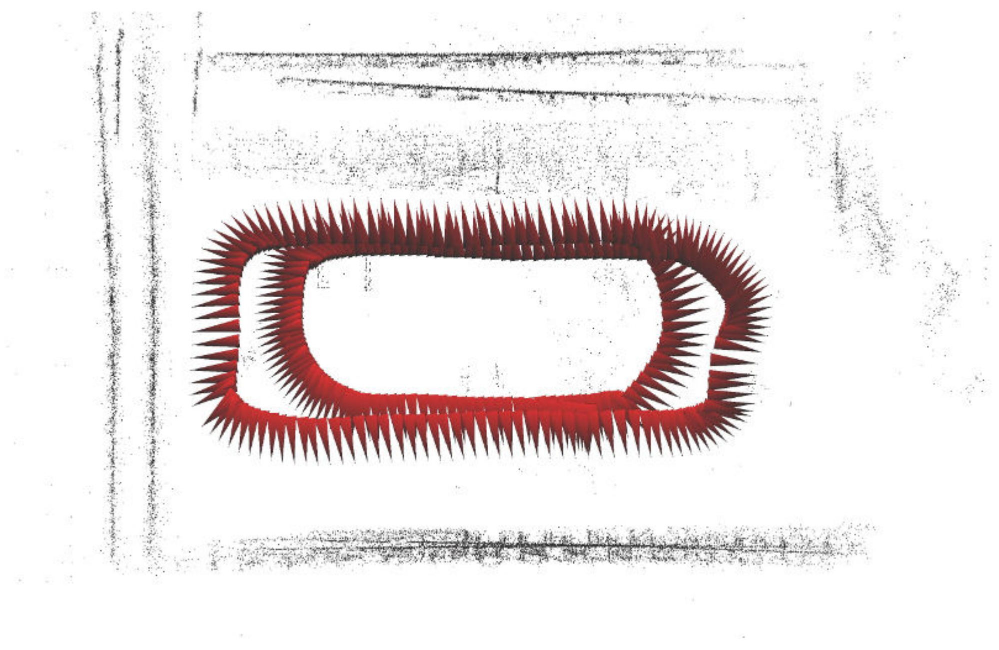


Figure 6.5: Sparse point cloud with camera poses after initial path computation.

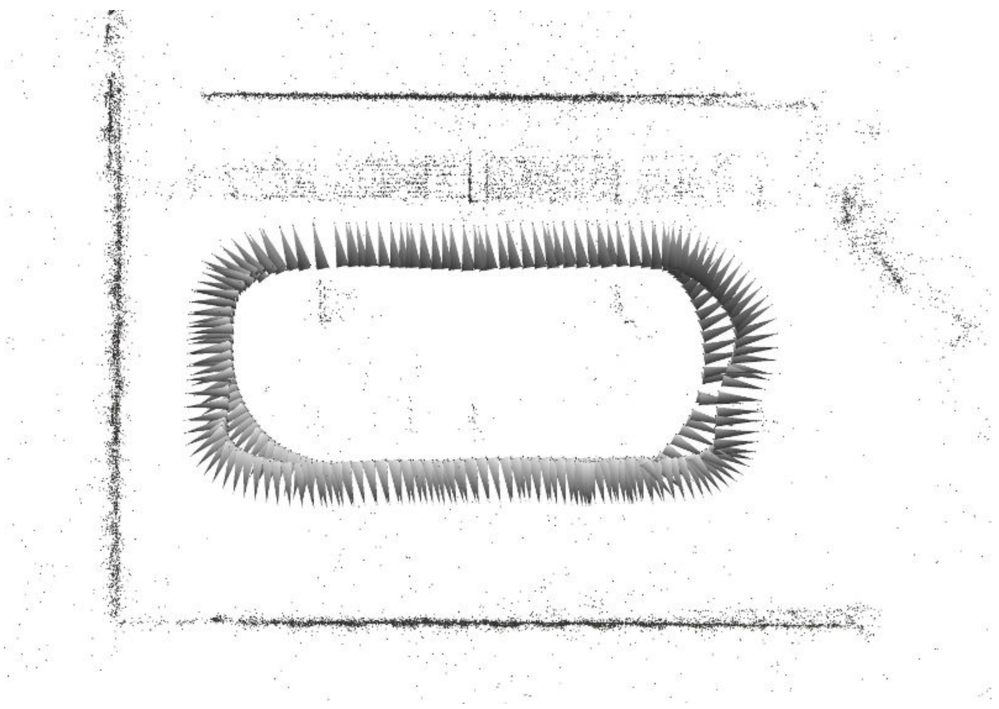


Figure 6.6: Sparse point cloud with camera poses after loop closure matched.

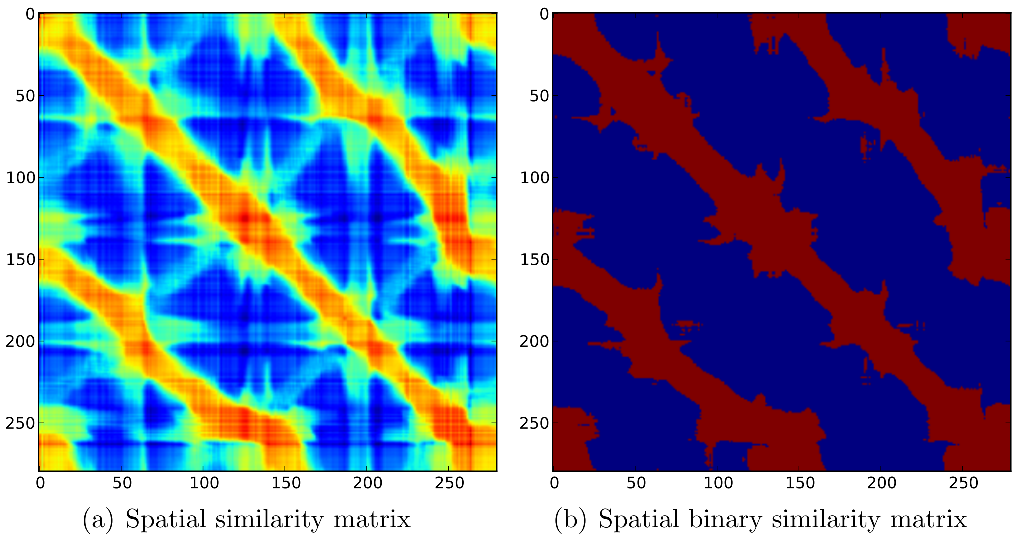


Figure 6.7: Spatial similarity matrix for loop closure.

in the graph two times with different frame ids. First, an image is included by a triplet at the beginning of the loop and a second time at the end of the loop. A factor is computed which describes the impact that the closing transformation has on a certain image.

$$s_f = \begin{cases} 0 & : f \leq f_B \\ \frac{f - f_B}{f_E - f_B} & : f_B < f < f_E \\ 1 & : f_E \leq f \end{cases} \tag{6.6}$$

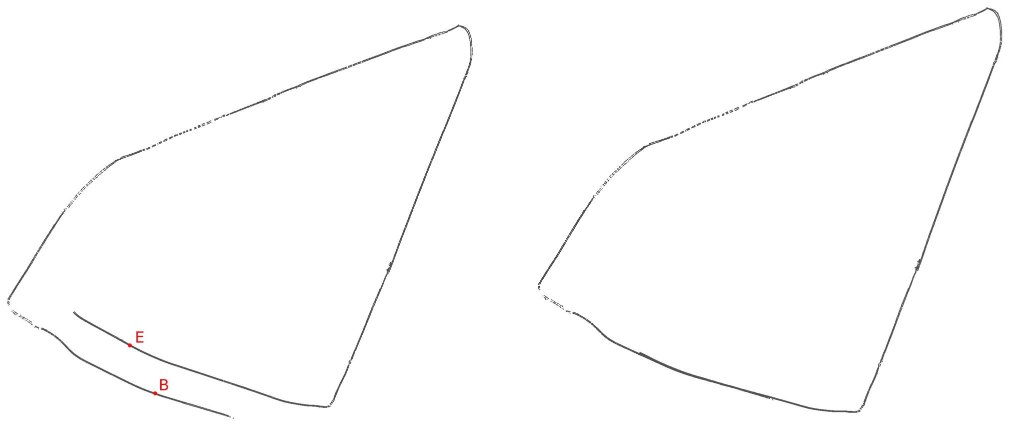
The closing transformation transforms camera f_E onto camera f_B by means of position and orientation. The closing transformation consists of a rotation $\mathbf{q}_{EB}$ and a translation $\mathbf{t}_{EB}$.

$$\mathbf{q}_{EB} = \mathbf{q}_B^{-1}\mathbf{q}_E \tag{6.7}$$

$$\mathbf{t}_{EB} = \mathbf{t}_B - \mathbf{t}_E \tag{6.8}$$

This transformation needs to be interpolated by the factor s_f computed in Equation 6.6. Several approaches to interpolate a rotation have been discussed in [17]. The one applied in this work uses a linear approximation over the rotation angle α, which results in a constant angular velocity. From $\mathbf{q}_{EB}$ the angle α as well as the rotation axis $\mathbf{v}$ can be extracted.

$$\alpha = 2\cos^{-1}(\mathbf{q}_{EB}[0]) \tag{6.9}$$



(a) Initial Path (b) Closed Path

Figure 6.8: Closure of large gap in loop with 10000 cameras.

$$\mathbf{v} = \frac{\mathbf{q}_{EB}[1:4]}{||\mathbf{q}_{EB}[1:4]||} \tag{6.10}$$

A new quaternion is defined for each camera which rotates the camera into the new coordinate system.

$$\mathbf{q} = [\cos(\alpha s_f/2), \mathbf{v}^T \sin(\alpha s_f/2)]^T \tag{6.11}$$

The new position and orientation of each camera is defined as follows:

$$\tilde{\mathbf{t}}'_f = \mathbf{q}(\tilde{\mathbf{t}}_f - \tilde{\mathbf{t}}_E)\mathbf{q}^{-1} + \tilde{\mathbf{t}}_E + s\tilde{\mathbf{t}}_{EB} \tag{6.12}$$

$$\mathbf{q}'_f = \mathbf{q}^{-1}\mathbf{q}_f \tag{6.13}$$

This procedure can be repeated if several loop closures within the dataset are detected. The newly defined solution is further optimized in a BA step. This time, the point correspondences in the loop will fit together and are not identified as outliers. The final BA is also necessary to adjust displacements which were erroneously included in the closing procedure since the gap of the loop was initially not linearly distributed on all cameras between camera f_B and camera f_E. An example in which the initial path had a large error which could be closed is given in Figure 6.8.

CHAPTER VII

Results

This thesis describes a novel SfM toolchain with the aim to increase robustness and accuracy. The following chapter evaluates the performance of the newly presented approach. The evaluation is done on different datasets to show the universality of the pipeline. Special attention is paid to the accuracy and precision. If the ground truth is given, such as for the FTS and the Castle-P30 dataset, the global accuracy will be analysed by mapping the reconstruction to the ground truth with a Helmert transformation. Else, the precision of the dataset is evaluated by comparing multiple measurements. For each dataset, different aspects are selected which are analysed in detail. Different remarks are given concerning what is necessary for an accurate reconstruction such as reliable feature matches. It will be shown that the newly presented inverse BA is often able to converge even in cases where BA does not. Visual evaluation of the resulting sparse point cloud can give a qualitative assessment, but does not provide an objective and measurable performance criterion. The reprojection error is a measurable indicator for the quality of a reconstruction, but as it depends on the individual judgment of outliers, it is not an objective and unbiased performance criteria. The only mostly unbiased way to evaluate the performance of a reconstruction is the comparison to ground truth. If this is not possible, comparing multiple reconstructions against each other is the next best possibility. The datasets used for evaluation are chosen from different scenarios.

1. The FTS dataset is acquired in the lab with constant light conditions. The object of interest has a small size of 96 cm. It is used to evaluate the accuracy of the system in comparison to GCP. This dataset is taken also for evaluating the precision by comparing the results of multiple reconstructions of the same object.

2. The Castle-P30 dataset is publically available with a given ground truth [85]. It shows a courtyard with a medium size of approximately 50-60 m.

3. The KPM dataset is aquired with a camera mounted on a Segway. The captured area has a size of approximately 2400 m^2. The captured area is surrounded two times to have enough data to evaluate the precision by merging two reconstructions as well as evaluating the loop closure approach.

4. The Car dataset shows that the toolchain is able to handle even a dataset with 10^4 images. The images are extracted from a video sequence acquired with a low cost outdoor camera mounted on a car.

5. The UAV dataset shows a house which was reconstructed from only 145 images.

Many more datasets have been successfully reconstructed but, only a representative set is seleceted to show the generality of the algorithms.

7.1 Faller Train Station

The FTS dataset was acquired under lab conditions. The object of interest is a model house made by Faller® representing the train station Reichenbach. It has an overall size of 960 × 400 × 665 mm. For evaluation, 90 points on the object are measured with a Leica TCRA 1103 tachymeter. The coordinates of the measured 3D points are optimized in a network adjustment [47] as shown in Figure 7.1. Each 3D point is connected to manually assigned image coordinates. According to the network adjustment, the GCP are measured with an average accuracy of approximate $\sigma = 0.7$ mm. 200 images are taken around the object with a Panasonic GF1, 20 mm prime lens, in full resolution (4000 × 3000). The indexed points are marked within 15 images well distributed over the dataset. An id is assigned to each point as visible in Figure 7.2 starting from id 100. As not all points could be computed with a high precision during a least squares network adjustment, the ids exceed the total number of points. The reconstructed camera path together with the sparse point cloud is visualized in Figure 7.5.

7.1.1 Loop Closure

When capturing the FTS images, the object was surrounded two times. The first circle captured the lower part of the object whereas the second captured the roof. Each circle produces one loop closure as visualized by the red pyramids in Figure 7.5. The first loop closure appears after the first full circle around the

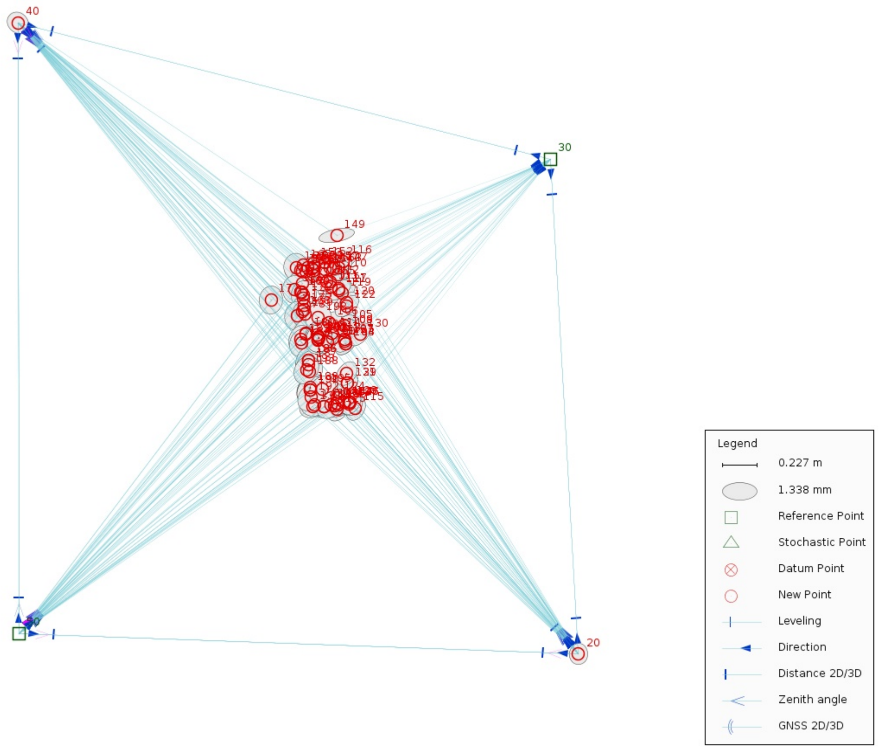


Figure 7.1: Ground truth measurement adjustment (JAG3D [47]).

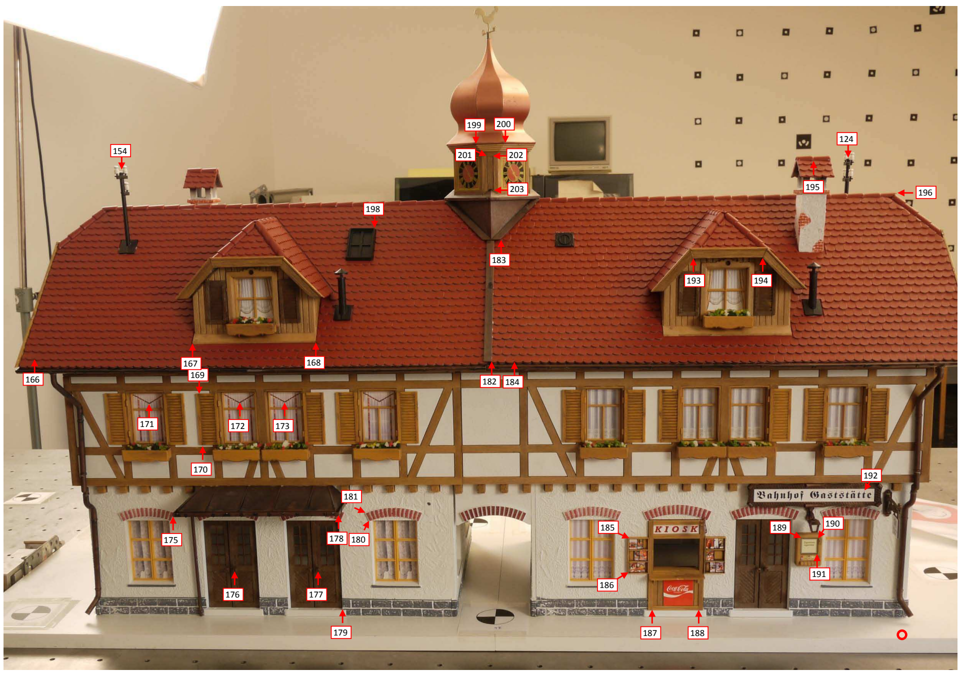


Figure 7.2: FTS with marked coordinates of GCP.

VDA	VOC
0.2	69.6

Table 7.1: Speed comparison of VDA and VOC in seconds.

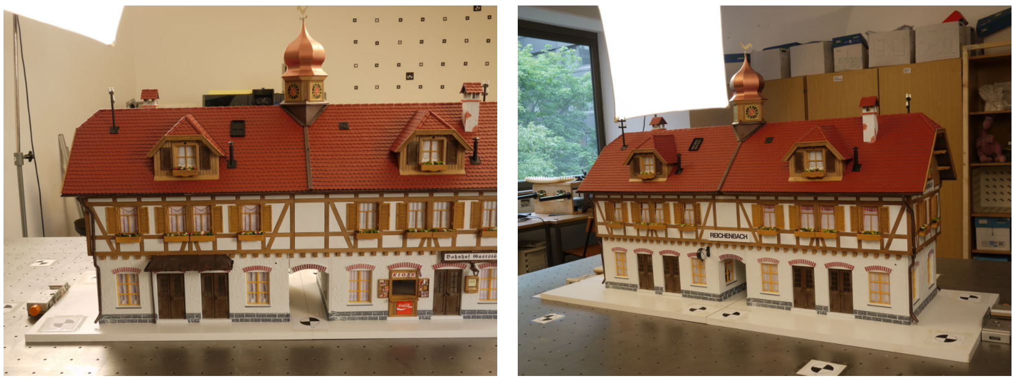
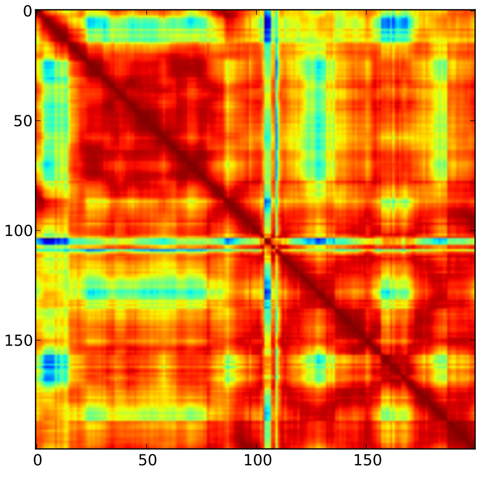
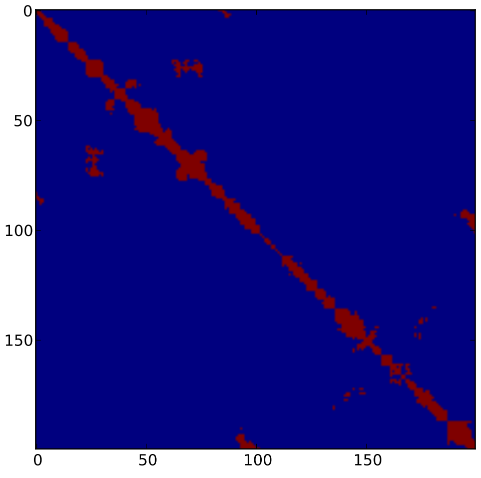

(a) Image 20 (b) Image 65

Figure 7.3: Example of a wrongly identified loop closure by VDA and VOC.

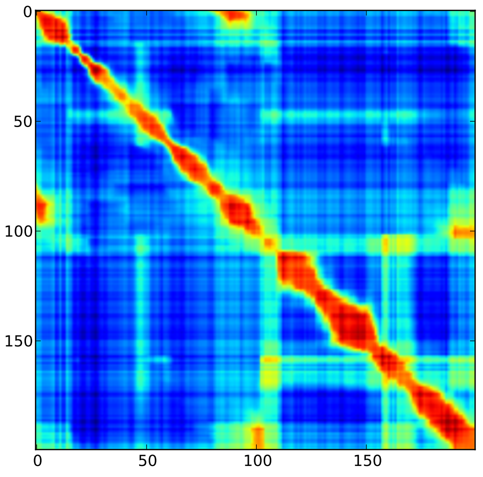
lower part of the object is closed, approximately between image 0 and image 85. The second loop closure is located at image 95, which has a strong overlap with image 198. It closes the loop which captured the roof of the train station. The symmetric structure of the train station is challenging for image based loop closure procedures. The two loop closures were successfully detected by VDA and the spatial loop closure strategy, as shown in Figure 7.4. The similarity matrices of the image based methods (Figures 7.4(a) and 7.4(e)) show a false loop closure between image 20 (Figure 7.3(a)) and image 65 (Figure 7.3(a)). This failure does not occur in the spatial loop closure method, but there is a wrongly identified closure between image 105 and image 160. The viewing angle between these two images is in the accepted range which was set empirically to 21.5 degrees, but the surface of the roof is not visible in the other image. Intersecting the binary appearance and spatial similarity matrix leads to the decision matrix in Figure 7.4(f). The newly presented VDA approach finds the same loop closures as estimated with the vocabulary tree, but with a smaller computational effort. The time comparison shown in Table 7.1 was computed on an Intel i7 with 3.07 GHz. Only one CPU was used to make a fair comparison. All loops were found with no false positive detection by intersecting the spatial and visual loop closure detection.



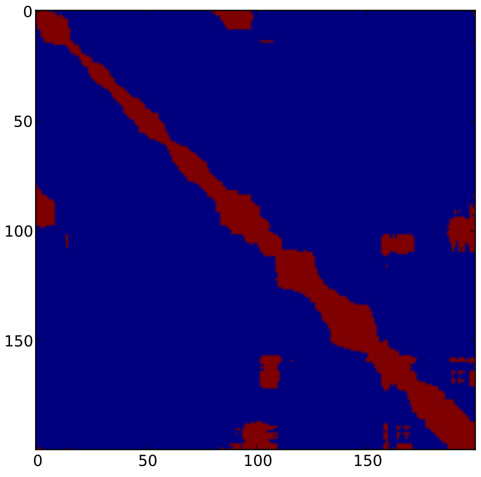
(a) Appearance similarity matrix (VDA)



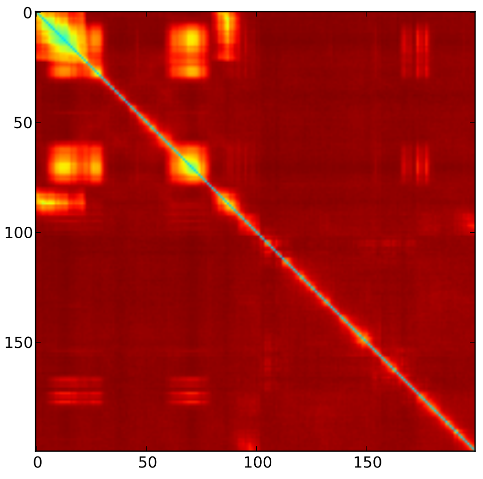
(b) Binary similarity matrix (VDA)



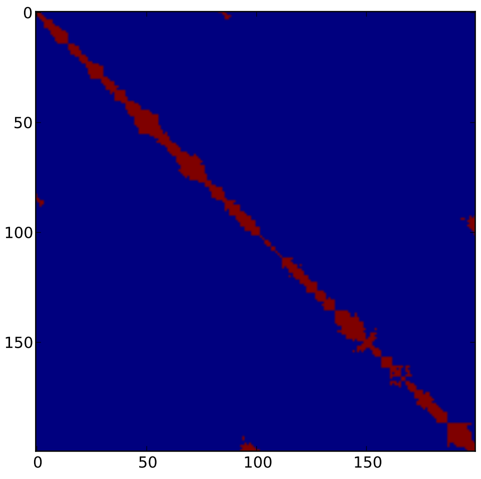
(c) Spatial similarity matrix



(d) Spatial binary similarity matrix



(e) Appearance similarity matrix (VOC)



(f) Intersection of Figs. 7.4(b) and 7.4(d)

Figure 7.4: Loop closure analysis of the FTS dataset.

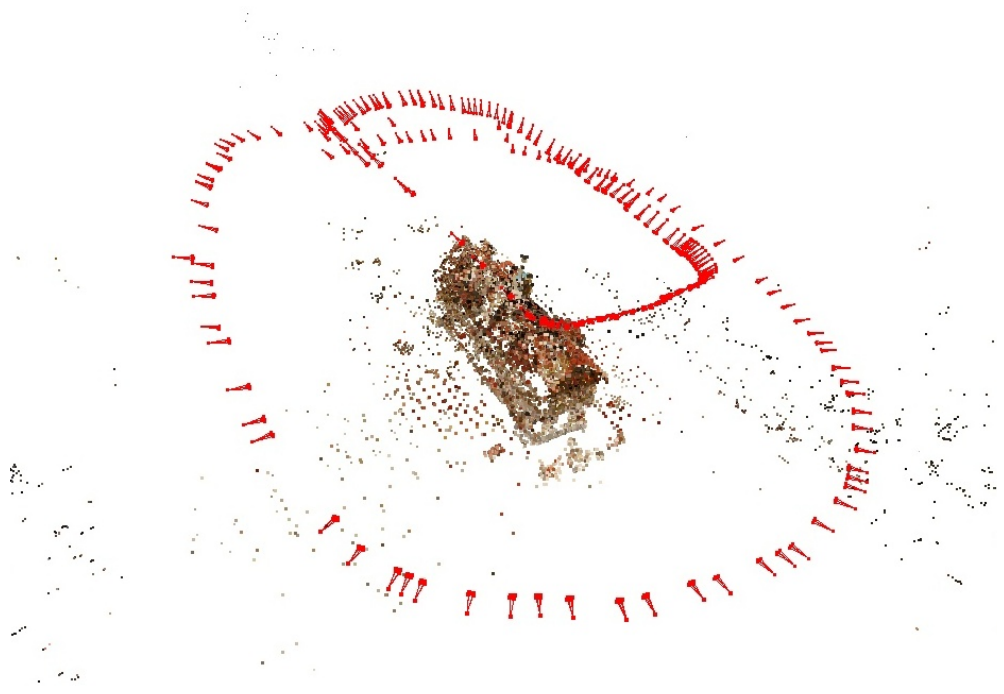
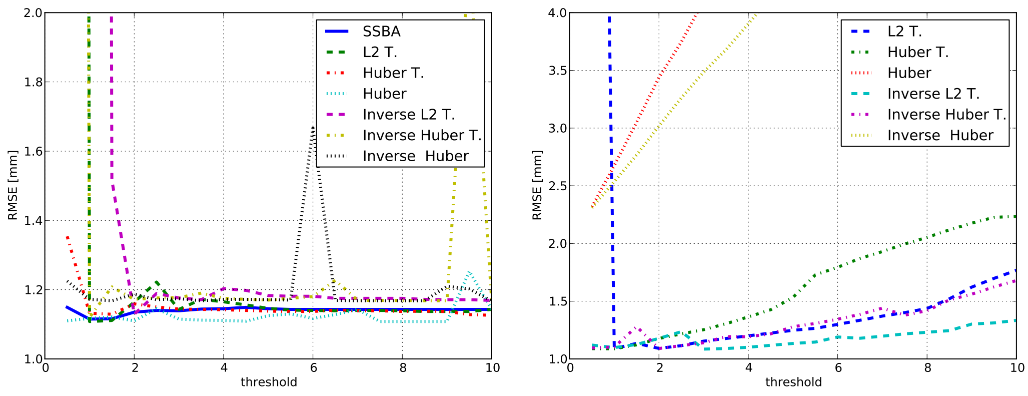


Figure 7.5: Reconstructed camera path with sparse point cloud.

7.1.2 Accuracy

From the reconstructed path, the manually marked correspondences are triangulated. Since the reconstruction is unique up to seven parameters, a Helmert transformation is computed which transforms the reconstruction to the GCP. As BA is an important aspect of the reconstruction, the initial path, which was generated as described in Section 5.4.2, is subsequently optimized with different robust cost functions, which is parametrized by the threshold in Tables 7.2 and 7.3 (see Figure 2.5 for different cost functions). Each bundled path is used to triangulate the manually assigned image coordinates which correspond to the measured GCP. The sparse set of points is then transformed to the ground truth as described in Section 2.5, and the resulting differences are listed in Table 7.2 and summarized in Figure 7.6(a). The same evaluation is repeated (Table 7.3 and Figure 7.6(b)), but this time the global ids are generated from all feature matches (only epipolar filter applied) and not only the matches which were filtered by a camera triplet. This time the global ids contain much more outliers, which strongly influences the standard BA parametrization.

The choice of a robust cost function with its parameters is in practice not an easy task. The threshold in pixel is chosen in a wide range to present the impact the parameter has on the convergence of the different methods. For the optimal



(a) Filtered feature matches (b) All feature matches

Figure 7.6: RMSE of the FTS dataset for different BA configurations corresponding to Tables 7.2 and 7.3.

		OpenOF			OpenOF Inverse		
Thresh. in Pixel	SSBA	L2 T.	Huber T.	Huber	L2 T.	Huber T.	Huber
0.5	1.149	32.232	1.354	1.110	36.738	19.552	1.225
1.0	1.115	1.108	1.132	1.117	25.824	1.120	1.172
1.5	1.116	1.111	1.129	1.121	1.517	1.208	1.168
2.0	1.135	1.164	1.159	1.110	1.129	1.172	1.187
2.5	1.140	1.222	1.150	1.145	1.189	1.176	1.171
3.0	1.139	1.143	1.144	1.115	1.176	1.177	1.174
3.5	1.144	1.172	1.143	1.112	1.171	1.190	1.168
4.0	1.145	1.165	1.142	1.111	1.203	1.175	1.172
4.5	1.149	1.156	1.140	1.109	1.198	1.172	1.172
5.0	1.145	1.146	1.138	1.125	1.183	1.170	1.171
5.5	1.143	1.141	1.137	1.130	1.181	1.175	1.170
6.0	1.143	1.140	1.137	1.117	1.181	1.180	1.667
6.5	1.143	1.139	1.141	1.128	1.175	1.227	1.169
7.0	1.143	1.140	1.139	1.143	1.175	1.169	1.168
7.5	1.143	1.140	1.139	1.108	1.175	1.169	1.168
8.0	1.143	1.139	1.137	1.108	1.175	1.169	1.168
8.5	1.143	1.137	1.138	1.108	1.172	1.168	1.168
9.0	1.143	1.137	1.137	1.108	1.171	1.168	1.210
9.5	1.143	1.137	1.128	1.253	1.171	2.167	1.202
10.0	1.143	1.143	1.126	1.143	1.169	1.167	1.167

Table 7.2: RMSE in mm compared to ground truth with filtered feature matches (filter from Section 4.1.4.3). The truncation of Huber T. is applied at twice the threshold.

		OpenOF			OpenOF Inverse		
Thresh. in Pixel	SSBA	L2 T.	Huber T.	Huber	L2 T.	Huber T.	Huber
0.5	2.043	14.095	1.089	2.318	1.120	1.095	2.309
1.0	472.328	1.089	1.088	2.676	1.098	1.103	2.536
1.5	469.797	1.135	1.122	3.050	1.125	1.280	2.767
2.0	378.866	1.091	1.177	3.438	1.178	1.090	3.018
2.5	386.183	1.115	1.217	3.745	1.236	1.115	3.261
3.0	377.273	1.154	1.253	4.105	1.085	1.140	3.485
3.5	377.223	1.177	1.305	4.465	1.090	1.193	3.679
4.0	380.081	1.200	1.364	4.905	1.101	1.192	3.903
4.5	377.184	1.224	1.426	5.217	1.118	1.217	4.132
5.0	377.195	1.248	1.533	5.542	1.134	1.279	4.339
5.5	377.226	1.265	1.728	5.922	1.146	1.307	4.550
6.0	3.738	1.299	1.794	6.233	1.190	1.344	4.804
6.5	377.767	1.335	1.871	6.538	1.179	1.386	4.975
7.0	4.118	1.371	1.930	6.816	1.197	1.440	5.171
7.5	377.370	1.401	1.996	7.126	1.217	1.379	5.360
8.0	4.441	1.437	2.053	7.392	1.231	1.418	5.571
8.5	4.663	1.521	2.118	7.655	1.245	1.511	5.715
9.0	4.760	1.622	2.176	7.892	1.302	1.561	5.914
9.5	4.931	1.698	2.227	8.067	1.312	1.624	6.046
10.0	5.088	1.768	2.234	7.176	1.334	1.679	6.213

Table 7.3: RMSE in mm compared to ground truth with all feature matches (only epipolar filter applied). The truncation of Huber T. is applied at twice the threshold.

solution the outlier threshold might be small (less than a pixel) but setting the threshold initially to a small value can cause the optimization not to close a loop as the measurements of the images forming the loop are treated as outliers.

The SSBA implementation does not converge for many different thresholds as shown in Table 7.3 by the high RMSE. While all BA approaches worked reasonably well for the filtered case, the novel inverse depth BA mostly outperforms the other approaches for the unfiltered case.

Outliers have less impact on the overall solution, due to the smaller degrees of freedom of inverse BA over standard BA. OpenOF L2 T. should be compared to OpenOF Inverse L2 T. etc. As shown in Figure 7.6 the truncated robust cost functions perform better in case of outliers, where the Huber norm performance better on the filtered feature matches. Applying strict outlier filters is mostly a good choice, but also correct feature matches are sometimes filtered out. Comparing Tables 7.2 and 7.3 shows that on average the filtered feature matches perform better, but the most accurate result was found with all feature matches. The reconstructed path shows a high accuracy with a RMSE of 1.085 mm for the GCP. The accuracy is close to the accuracy of the ground truth. It will be more and more complicated to generate a ground truth which has to be a magnitude better than the system which should be evaluated.

7.1.3 Precision

In case no ground truth is given, the quality of a reconstruction can be evaluated by reconstructing the object of interest from two datasets. These two reconstructions are merged as described in Section 5.4.4. Each reconstruction contains approximately 25000 sparse points. A correspondence was found for 1100 points with a SIFT matching ratio of 0.6. It is assumed that many of these correspondences are outliers, due to the symmetric structure of the object. The Helmert transformation is computed in two steps: First, all 1100 correspondences are used to compute the transformation with a robust L2 truncated cost function. The threshold, which is applied for the difference in one dimension, is decreased in 3 steps to 0.005 m. Second, only the points which have an error of less than 5 mm (ca. 62 %) are used to refine the calculated transformation. The error of the points are sorted in increasing order. Figure 7.7 shows the results for all 1100 points. The best 40 % of the points have an error of less than 1.68 mm. This value coincides with the accuracy computed in the last section. The last 30 % can be interpreted as outliers.

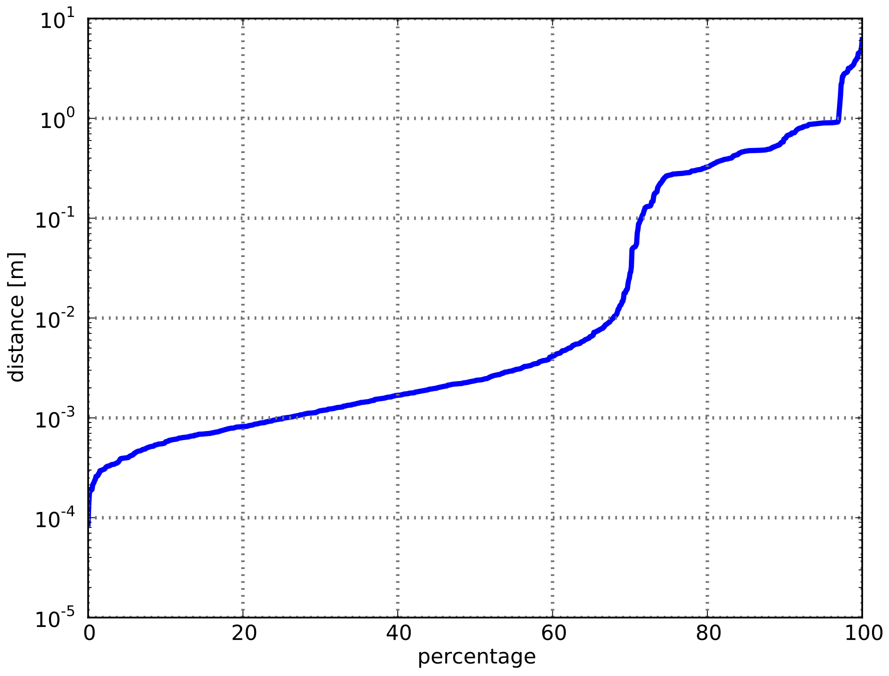


Figure 7.7: Precision of the FTS dataset.

7.2 Castle-P30

The Castle-P30 dataset from [85] comes with 30 images at a resolution of 3072×2048 (see Figure 7.8). It captures the court yard of the Ettlingen-Castle. The dataset comes along with the internal as well as the external camera parameter. The ground truth is established with a LIDAR system by measuring targets which are also visible within the images. [85] states a position accuracy of the ground truth for the Herz-Jesu-R23 dataset of 1-2 cm. A similar accuracy for the Castle-P30 dataset is assumed, even though the covered area of the Castle-P30 dataset is much larger in comparison to the Herz-Jesu-R23. The maximum distance of two cameras in the dataset is 44.97 meters. With only 30 images of the court yard, the dataset is relative sparse. In most cases only three images overlap. The dataset is reconstructed with the presented pipeline, as well as with VisualSFM [97]. Both pipelines successfully reconstructed the dataset. The computed camera positions are transformed with a Helmert transformation to ground truth. The position error and the orientation error of the cameras are plotted in Figure 7.9. The corresponding MAE and RMSE are shown in Table 7.4. The new developed inverse BA approach performs significantly better than BA and VisualSFM.

Figure 7.8: Sample image of the Castle-P30 dataset.

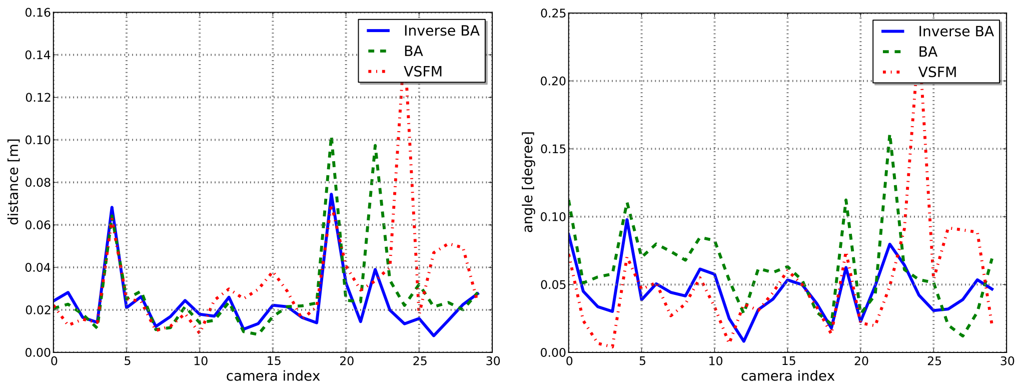


Figure 7.9: Distance and angle error of the Castle-P30 dataset.

	Inverse BA	BA	VSFM
Position MAE [m]	0.0231	0.0272	0.0328
Position RMSE [m]	0.0273	0.0348	0.0416
Orientation MAE [°]	0.0457	0.0619	0.0510
Orientation RMSE [°]	0.0496	0.0695	0.0661

Table 7.4: MAE and RMSE of Inverse BA, BA and VisualSFM.

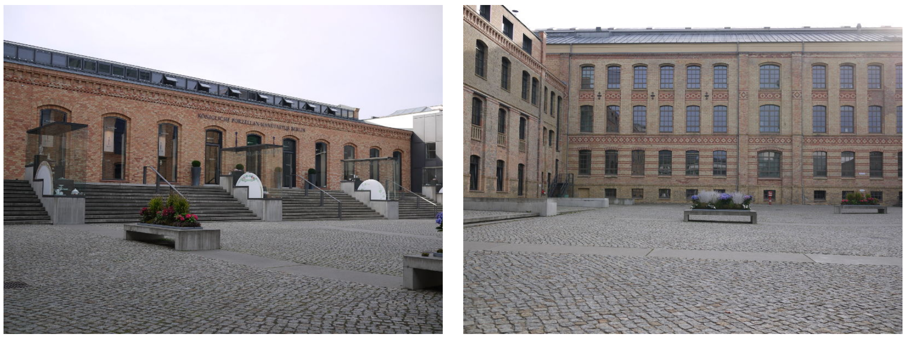
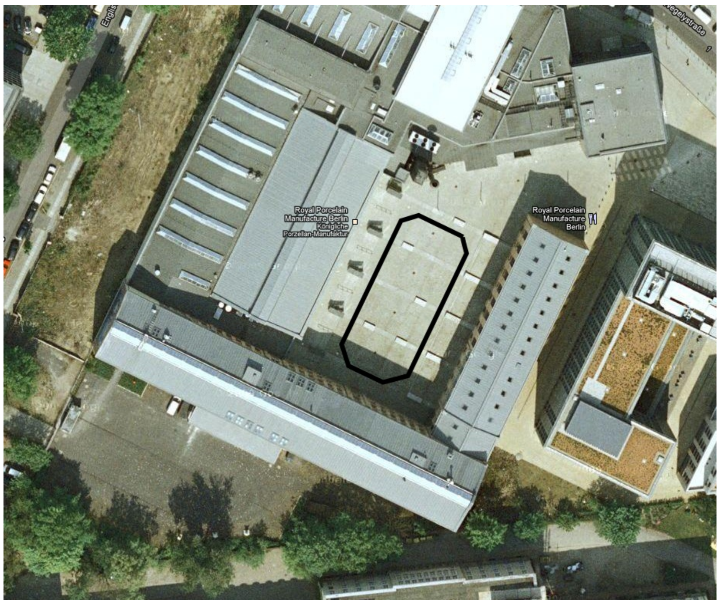


Figure 7.10: Sample images of the KPM dataset.


Royal Porcelain Manufacture Berlin Königliche Porzellan-Manufaktur
Royal Porcelain Manufacture Berlin
Figure 7.11: Top view of the court yard of KPM [25] with an approximate camera path in black.

7.3 KPM

The KPM dataset was acquired at the factory of *Köngliche Porzellan-Manufaktur* in Berlin. The dataset consists of 280 images taken with a Panasonic GF1, 20 mm prime lens. In contrast to the FTS dataset, the image resolution is reduced to 2815×2112 pixels. This reduction is necessary to acquire images with 3 frames per second (FPS) in the continuous shooting mode of the camera, otherwise the data transfer within the camera would not be capable of handling all incoming data and writing the images onto a memory card. The camera was mounted on a Segway. The captured area has a size of approximately 60×40 meters. Sample images as well as a top view of the area

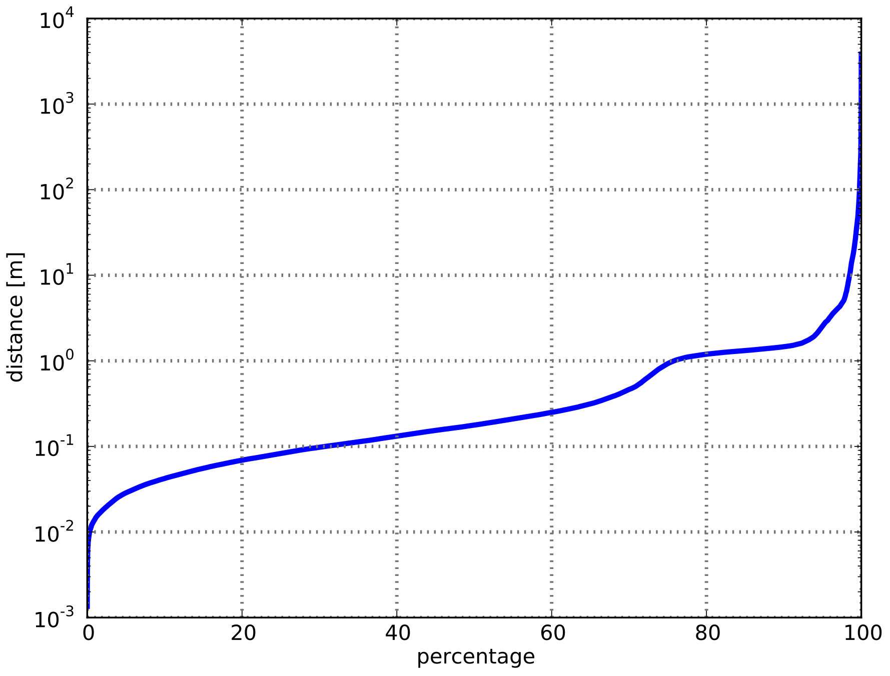


Figure 7.12: Precision of the KPM dataset.

are shown in Figures 7.10 and 7.11. The images were acquired by driving with a constant speed two times around the square to produce a high overlap for the loop closure procedure. The camera was mounted perpendicularly to the driving direction, capturing the buildings on the opposite side of the square. The loop is closed for the first time at frame 150 and for the second time at 270 as discussed in Chapter VI. As no ground truth of the dataset exists only the precision is evaluated by splitting the dataset into two equal sets of images. Therefore, the frames were assigned alternating to the first and the second dataset. The datasets were reconstructed individually. The resulting reconstructions are merged and the error of the merging procedure is used as evaluation measure indicating the precision of the two reconstructions. The distance of the best 40 % of the points is less than 13.6 cm (see Figure 7.12). Considering the covered size, the dataset is around 60 times larger than the FTS dataset. Multiplying the precision of the FTS dataset with the size results in $10.08\ cm = 60 \times 0.168\ cm$. The obtained precision is lower due to the lower image resolution. Nevertheless, the result is more than satisfying considering the size of the object.

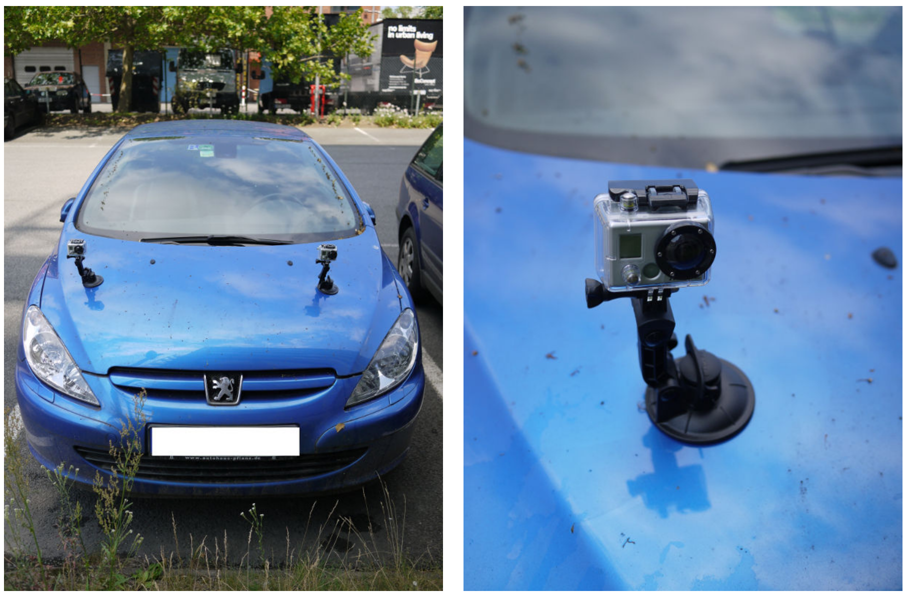


Figure 7.13: Acquisition setup with two GoPros on a car.

7.4 Car

This dataset was acquired with two GoPro HD HERO2 cameras mounted on a car as shown in Figure 7.13. Videos with a resolution of 1920×1080 pixels and 30 FPS were captured and converted to individual images. The cameras have a field of view of 170° and a strong radial distortion. The fish-eye camera model is used to undistort the images. A total of 8100 images were extracted from each video, but only a subset of 5000 images was selected for reconstruction. Areas where many similar or identical images exist, e.g. when the car stopped at a red traffic light, were thinned out. The path has a total length of about 1.9 km with an overlap of 200 m. The video streams were synchronized manually up to 1/60 sec. For the reconstruction, the streams are assumed to be independent. The GPU based inverse BA implementation is limited by the amount of memory of the graphics card. For this reason, the graph was separated in overlapping chunks of 400 cameras. The camera path is reconstructed for each chunk and optimized by inverse BA. For each chunk, the mean distance of all synchronized image pairs is scaled to an absolute scale which was measured manually. The accumulated drift of the path is shown in Figure 6.8. The loop was closed with the approach presented in Section 6.3. Both, the driven and the reconstructed path are shown in Figure 7.14. This figure was created by manually overlaying the reconstructed path on the image from Google maps adjusting the scale and

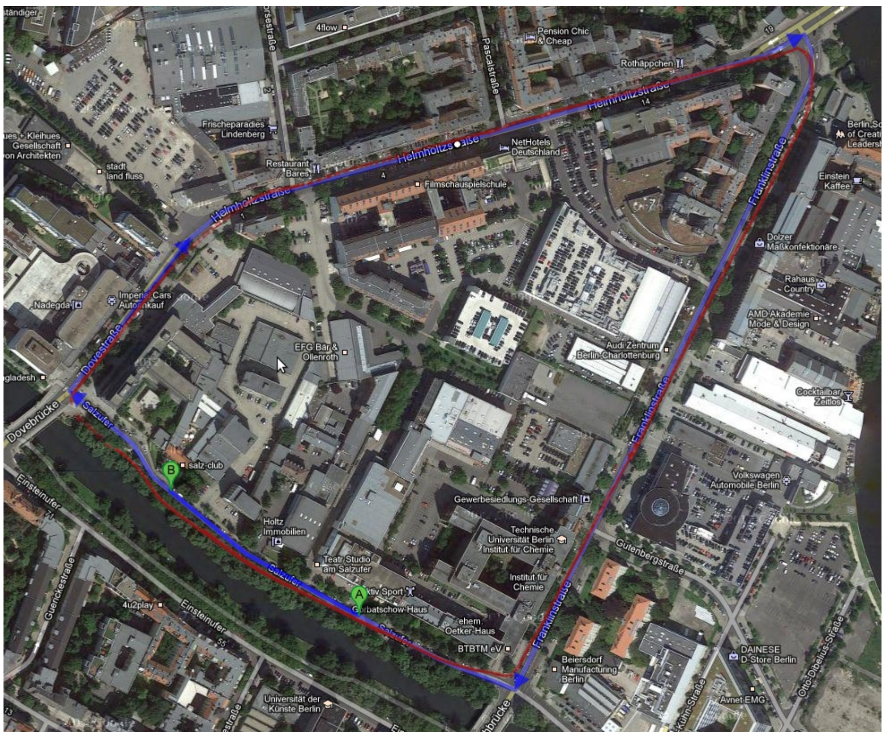


Figure 7.14: Driven path in blue, reconstructed path in red [25].

rotation. The overlay of the calculated path and the map fit accurately. In summary, the presented approach works even for large scale reconstructions with several thousand images. Incorporating further information such as a true stereo constraint or additional sensors, e.g. Global Positioning System (GPS), could make the results even better.

7.5 UAV

In contrast to the Segway and the car, where the movement is restricted to a near planar surface, the UAV has the ability to capture images with full degrees of freedom. This dataset consists of 145 images. The camera (Panasonic GF1) is mounted on a Falcon 8 (see Figure 1.1) from Ascending Technologies. Because the payload is specified with 500 g, it is impossible for the UAV to carry a suitable laserscanner, but it can carry a high quality compact system camera. Sample images of the dataset are given in Figure 7.15. Figure 7.16(a) shows the calculated camera path according to the workflow developed in this thesis. The path is not regular, reflecting the windy conditions during the flight. Other approaches, which are based on motion models, would probably fail under such

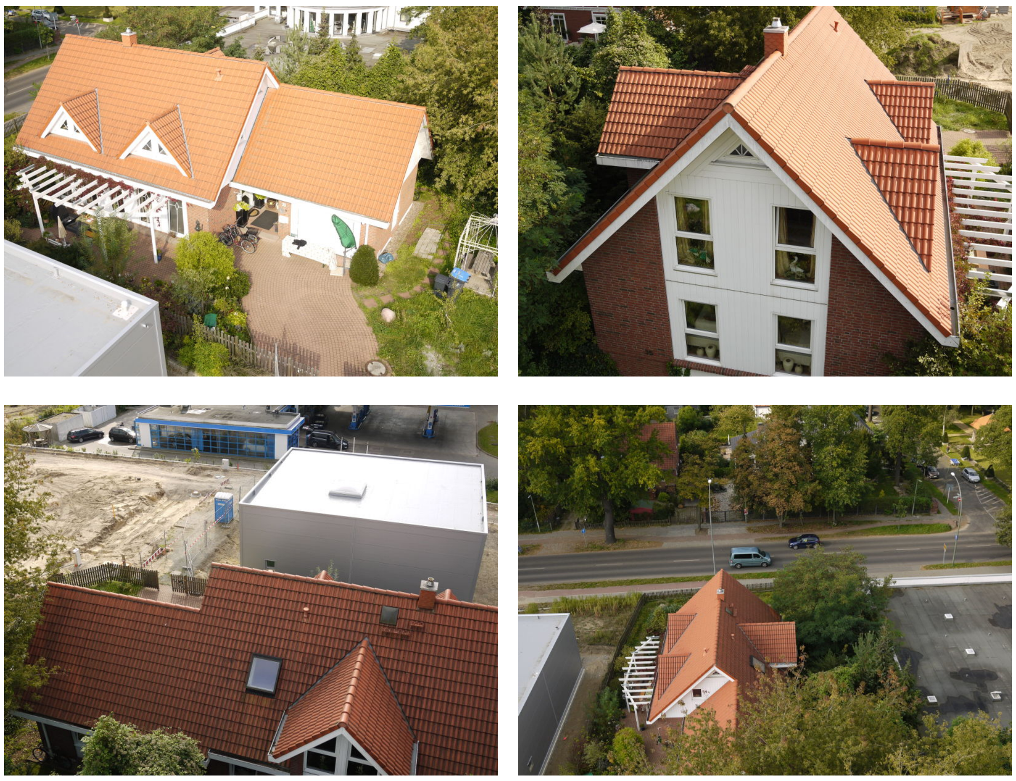
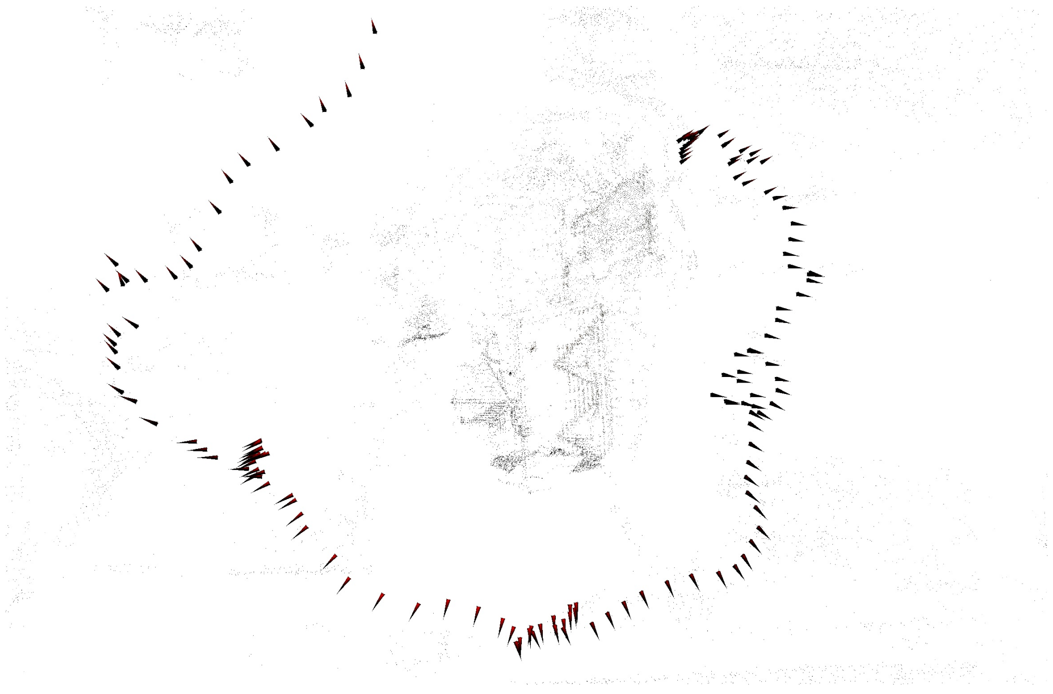


Figure 7.15: House acquired by a UAV.

circumstances. From the camera path a quasi dense point cloud is computed utilizing Patch-based Multi-view Stereo (PMVS) [24]. The point cloud in Figure 7.16(b) shows many details even in areas which do not belong to the house such as a street light. The amount of details indicate a well reconstructed camera path.

In summary, it could be shown, that the approach which is proposed in this thesis gives good results for a variety of different examples. The datasets were chosen from different scenarios with completely different scales. Camera paths with a length of a few meters up to 1.9 km have been reconstructed. The presented approaches outperform state of the art methods in either robustness, speed or accuracy.


(a) Camera Path

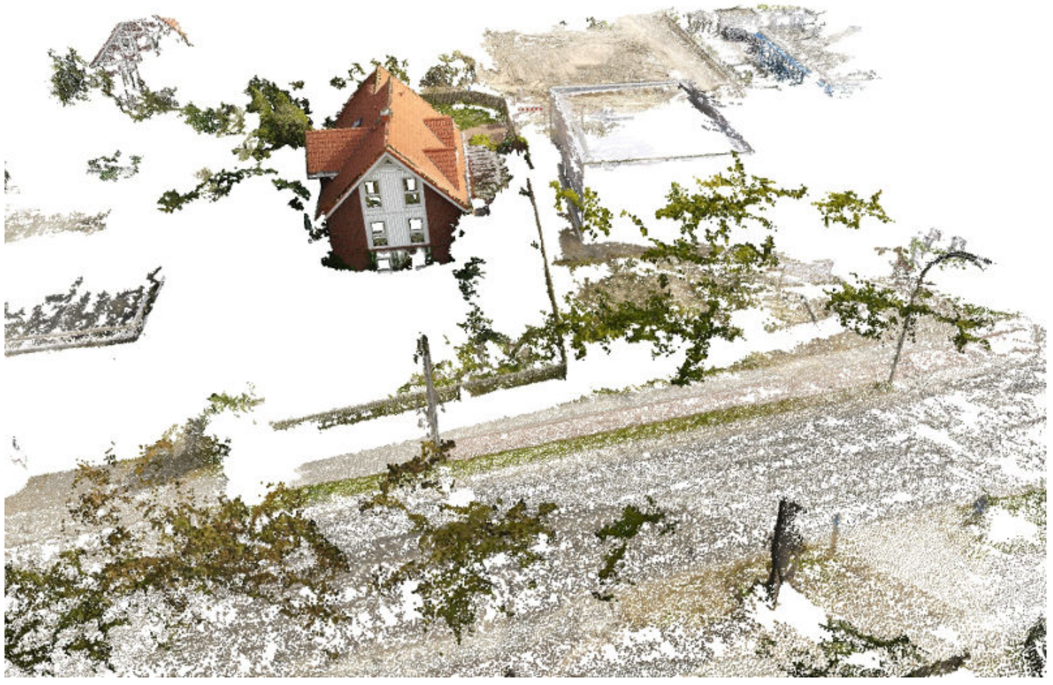

(b) Quasi Dense Point Cloud

Figure 7.16: Camera path and quasi dense point cloud for the UAV dataset.

CHAPTER VIII

Conclusion and Outlook

Within this thesis a modular approach for solving the SfM problem has been presented. The SfM problem can be subdivided into different modules. Improvements in either robustness or computational speed have been introduced at different stages of the toolchain. Data handling, feature estimation, geometry computation and non-linear optimization are the most important modules, which are needed for a robust 3D reconstruction system. The center of the approach is a well designed data structure, which every module of the toolchain can access and update, thus simplifying the integration of new modules or the exchange of existing modules. The relational data model is suitable for the data, which is computed in each step of the toolchain. Currently, the main limitation when computing a 3D reconstruction by several clients lies in the performance of the MySQL database. Performance might be further improved by choosing a distributed database system optimized for serving massive amounts of data to different clients. Each client can process different sets of images, which makes the approach capable of being used in a cloud.

The second important part of the toolchain is the feature extraction. It includes the detection of interest points with a high repeatability rate. Furthermore, a distinctive description is assigned to a feature to find a match in a set of thousands of other features. With TBH a new matching method is presented, which was demonstrated to be superior to KD-trees and BBF. The main challenges in 3D reconstructions occur if the images taken by the users are either too sparse, resulting in a small overlap, or the viewing angle between two views changes too much. A perfect feature detector and descriptor is not invented yet. Great improvements can be expected if it is possible to match images with a large view point difference. Currently, the best available solution is SIFT which is incorporated in the presented toolchain.

Current research in SfM assumes static scenes. Dynamic parts are detected as outliers and filtered out. In case a moving object covers large parts of the image,

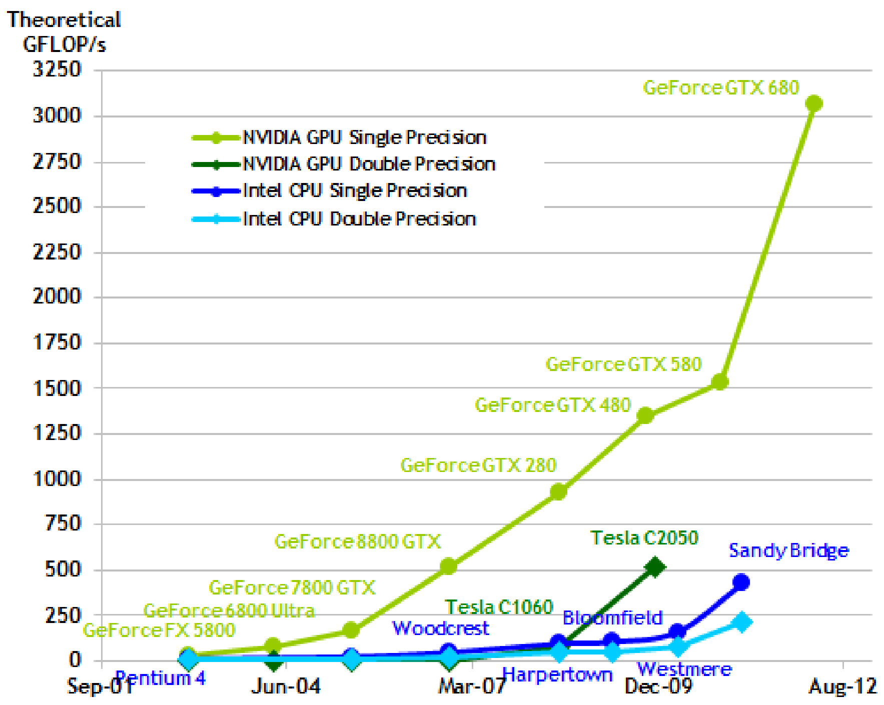


Figure 8.1: Computational power comparison between GPUs and CPUs [68].

potentially the surrounding scene is filtered out. When acquiring images with a car, similar challenges occur, e.g. when a bus drives in front of the car and the back of the bus may cover more than 50 % of the image.

So far, all SfM approaches are based on distinct interest points, but lately a new approach which considers every pixel in an image has been presented in the SLAM community [50, 64]. This is possible due to the high processing power of GPUs (see Figure 8.1 for a comparison). Incorporating every pixel of an image in a BA step can increase the precision by an order of a magnitude and directly results in a dense reconstruction. The dense point cloud and the camera positions are optimized at the same time. The relatively small amount of memory of consumer graphics cards is the main limitation.

A lack of experience in image acquisition or bad conditions can lead to missing or noisy data. In case of offline processing systems, these errors are not detected before processing the images. At that point in time, it might be costly or even impossible to return to the object and capture the missing parts. Computing a rough estimate of the object in realtime to detect if parts are missing, could avoid this problem. But the dependence between accuracy and time is not linear, it rather follows a step function. The number of iterations within RANSAC is the main parameter to adjust the runtime. In case the number of iterations is too low, the results will be completely wrong not following a normal distribution.

Instead of enabling the system to process bad data, improving the image acquisition is another approach. Simplifying the process of image acquisition can have a great impact when getting the technology in the consumer market. Instead of acquiring images, high resolution videos could prevent the following typical errors:

- Large viewpoint change.
- Only two images overlap instead of three.
- Capturing the scene as panorama with no baseline.

Most of these errors occur easily when taking images, because often the user moves to a position, makes several photos of the surrounding scene and then moves to a different position, mostly far away from other positions and repeats the process. However, when making a video, the user moves slowly around the object and intuitively performs the correct movement. With the video mode of digital single-lens reflex (SLR) cameras, high quality videos for 3D reconstructions can be made due to high quality lenses of the cameras. But, as shown by Strasdat et al. [84], a high number of features in contrast to a high framerate results in better accuracy. So it might be necessary to select an optimal set of key frames from the video.

Other research areas handle tracking of cars for video surveillance systems [13] or detecting and analysing people movements [93]. In both examples the camera is assumed to be static. Merging SfM with those areas could lead to interesting new applications, e.g. video surveillance with UAVs or driver assistance systems [73]. In this case both the camera as well as the objects of interest move. Reconstructing surrounding cars and predicting their pathways could be a promising application [76].

A logical extension handles not only moving but deformable objects [15]. Analysing the deformation of animals takes already part in current research. A camera system such as the quadrifocal camera setup in Figure 8.2 can be used to capture synchronous images of non-rigid objects such as animals. Each quadruple is reconstructed and the motion is analysed between multiple reconstructions. Once the surface of the object is reconstructed, a template is generated and the surface can be tracked by only one camera [61]. Considering multiple templates and multiple cameras, a complete non-rigid object can be modeled.

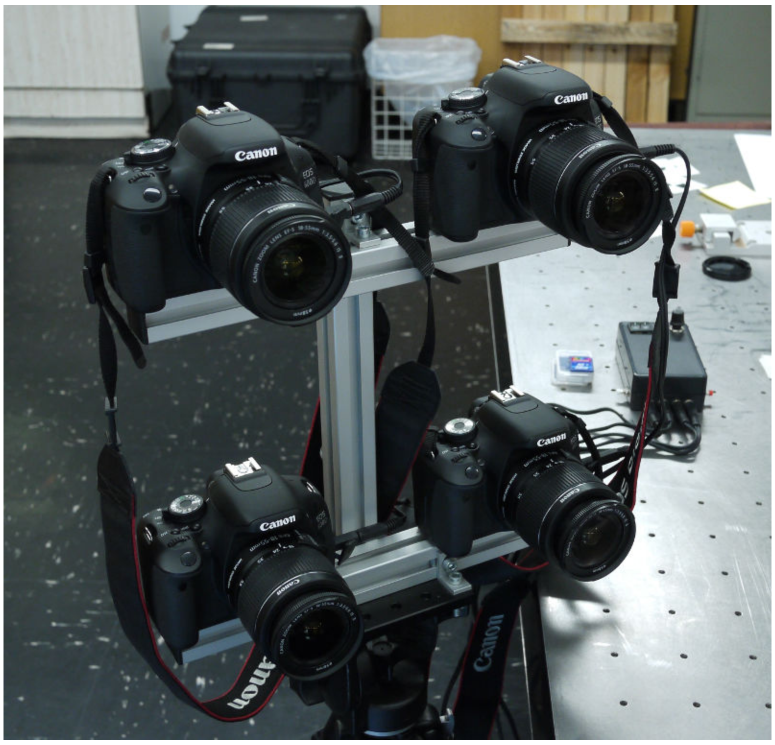


Figure 8.2: Quadrifocal camera setup to capture deformable objects.

Within this work complete SfM toolchain was developed, many challenging problems have been solved and new ideas have been presented. Especially, OpenOF can be a cornerstone for further research topics as it not only covers the interest of the computer vision community but can be used by everybody who has a non-linear least squares optimization problem.

Bibliography

[1] Sameer Agarwal, Noah Snavely, and Steven Seitz. Bundle adjustment in the large. In *ECCV*, pages 29–42. Springer, 2010.

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven Seitz, and Richard Szeliski. Building Rome in a day. In *ICCV*, pages 72–79. IEEE, September 2009.

[3] Motilal Agrawal. A Lie algebraic approach for consistent pose registration for general Euclidean motion. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 1891–1897. IEEE, 2006.

[4] Adrien Angeli, David Filliat, Stephane Doncieux, and Jean-arcady Meyer. A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 24(5):1027–1037, 2008.

[5] Jan Bartelsen, Helmut Mayer, Heiko Hirschmüller, Andreas Kuhn, and Mario Michelini. Orientation and dense reconstruction from unordered wide baseline image sets. *PFG Photogrammetrie, Fernerkundung, Geoinformation*, 2012(4):421–432, 08 2012.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *CVIU*, 110(3):346–359, 2008.

[7] Jeffrey Beis and David G. Lowe. Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In *CVPR*, pages 1000–1006. IEEE, 1997.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[9] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, pages 111–118. Omnipress, 2010.

[10] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *CVPR*, volume 1, pages 510–517. IEEE, 2005.

[11] Martin Byröd and Kalle Åström. Conjugate gradient bundle adjustment. In *ECCV*, pages 114–127, 2010.

[12] Javier Civera, Andrew Davison, and Martinez Montiel. Inverse Depth Parametrization for Monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, October 2008.

[13] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, 1998.

[14] Thomas Connolly and Carolyn Begg. *Database Systems, A Practical Approach to Design, Implementation, and Management.* Addison-Wesley, 2002.

[15] Tim Cootes. Deformable Object Modelling and Matching. In *ACCV*, pages 1–10. Springer, 2011.

[16] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties and Algorithms*, volume 46. Springer, 2007.

[17] Erik B. Dam, Martin Koch, and Martin Lillholm. Quaternions, interpolation and animation. Technical report, 1998.

[18] Ivan Dryanovski, William Morris, and Jizhong Xiao. An open-source pose estimation system for micro-air vehicles. In *ICRA*, pages 4449–4454. IEEE, 2011.

[19] Wolfgang Förstner. A Framework for Low Level Feature Extraction. In *ECCV*, volume 802, pages 383–394. Springer, 1994.

[20] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building Rome on a cloudless day. In *ECCV*. Springer, 2010.

[21] Jan-Michael Frahm and Marc Pollefeys. RANSAC for (Quasi-)Degenerate data (QDEGSAC). In *CVPR*, pages 453–460. IEEE, 2006.

[22] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[23] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[24] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *TPAMI*, 32(8):1362–76, August 2010.

[25] Google. Berlin - google maps, March 2012. http://maps.google.com.

[26] Ankur Handa, Richard A. Newcombe, Adrien Angeli, and Andrew J. Davison. Real-time camera tracking: When is high frame-rate best? In *ECCV*, pages 222–235, 2012.

[27] Matthew Harker and Paul O. Leary. First Order Geometric Distance (The Myth of Sampsonus). In *BMVC*, volume 1, pages 1–10. British Machine Vision Association, 2006.

[28] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.

[29] Olaf Hellwich, Cornelius Wefelscheid, Jakub Lukaszewicz, Ronny Hänsch, Adnan M. Siddique, and Adam Stanski. Integrated matching and geocoding of SAR and optical satellite images. In *IBPRIA 2013*. Springer, June 2013.

[30] Peter Huber. *Robust Statistics*, volume 1 of *Wiley Series in Probability and Statistics*. Wiley, 1981.

[31] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128, January 2011.

[32] Yekeun Jeong, David Nister, Drew Steedly, Richard Szeliski, and In-so Kweon. Pushing the Envelope of Modern Methods for Bundle Adjustment. In *CVPR*, pages 1474–1481. IEEE, 2010.

[33] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing*, 3(4):143–152, 2009.

[34] Fredrik Kahl and Richard Hartley. Critical Curves and Surfaces for Euclidean Reconstruction. In *ECCV*, pages 447–462. Springer, 2002.

[35] Juho Kannala and Sami S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *TPAMI*, 28(8):1335–1340, 2006.

[36] Yan Ke and Rahul Sukthankar. PCA-SIFT : A More Distinctive Representation for Local Image Descriptors. *Review Literature And Arts Of The Americas*, 2003.

[37] Sebastian Knorr, Matthias Kunter, and Thomas Sikora. Super-resolution stereo- and multi-view synthesis from monocular video sequences. In *3-D Digital Imaging and Modeling (3DIM 2007)*, Montral, Qubec, Canada, August 2007.

[38] Kevin Köser and Reinhard Koch. Perspectively invariant normal features. In *ICCV*, pages 14–21. IEEE, 2007.

[39] Ullrich Köthe and Michael Felsberg. Riesz-Transforms Versus Derivatives: On the Relationship Between the Boundary Tensor and the Energy Tensor. In *Scale Space and PDE Methods in Computer Vision*, volume 3459, pages 179–191. Springer, 2005.

[40] Erwin Kruppa. Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung. *Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw. Kl., Abt. Ila*, 122(122):1939–1948, 1913.

[41] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *ICRA*, pages 3607–3613. IEEE, 2011.

[42] Matthias Leuthäuser. Effizientes Matching von Feature-Deskriptoren in Echtzeit. Diplomarbeit, Technische Universität Berlin, 2010.

[43] Xiaowei Li, Changchang Wu, Christopher Zach, Svetlana Lazebnik, and J.M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, volume 8. Springer, 2008.

[44] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, volume 7572, pages 15–29. Springer, 2012.

[45] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pages 224–270, 1994.

[46] Hugh C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[47] Michael Lösler. JAG3D - Java Graticule 3D (OpenAdjustment) - The OpenSource Geodetic Network Adjustment Program, 2013. http://javagraticule3d.sourceforge.net/.

[48] Manolis I. A. Lourakis. Sparse non-linear least squares optimization for geometric vision. In *ECCV*, pages 43–56. Springer, 2010.

[49] Manolis I. A. Lourakis and Antonis A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, 36(1):1–30, 2009.

[50] Steven Lovegrove and Andrew J. Davison. Real-time spherical mosaicing using whole image alignment. In *ECCV*, pages 73–86. Springer, 2010.

[51] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, volume 2, pages 1150–1157. IEEE, 1999.

[52] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, November 2004.

[53] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, pages 674–679. Morgan Kaufmann Publishers Inc., 1981.

[54] Kaj Madsen, Hans Nielsen, and Ole Tingleff. Methods for Non-Linear Least Squares Problems (2nd ed.), 2004.

[55] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[56] Helmut Mayer, Jan Bartelsen, Heiko Hirschmüller, and Andreas Kuhn. Dense 3d reconstruction from wide baseline image sets. In *Proceedings of the 15th international conference on Theoretical Foundations of Computer Vision: outdoor and large-scale real-world scene analysis*, pages 285–304. Springer, 2012.

[57] J. Chris McGlone, Edward M. Mikhail, James Bethel, and Roy Mullen. *Manual of photogrammetry.* American Society for Photogrammetry and Remote Sensing, 2004.

[58] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *TPAMI*, 27(10):1615–1630, 2005.

[59] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *IJCV*, 65(1-2):43–72, 2005.

[60] Yang Mingqiang, Kpalma Kidiyo, and Ronsin Joseph. A survey of shape feature extraction techniques. *Pattern Recognition*, 2008(November):43–90, 2008.

[61] Markus Moll and Luc Van Gool. Optimal templates for nonrigid surface reconstruction. In *ECCV*, volume 7572, pages 696–709. Springer, 2012.

[62] Jean-Michel Morel and Guoshen Yu. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. *SIAM Journal on Imaging Sciences*, 2(2):438, 2009.

[63] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *Science*, 340(3):331–340, 2009.

[64] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, volume 1, pages 2320–2327. IEEE, 2011.

[65] David Nistér. An efficient solution to the five-point relative pose problem. *TPAMI*, 26(6), 2004.

[66] David Nistér and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, volume 2, pages 2161–2168. IEEE, 2006.

[67] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*, volume 43 of *Springer Series in Operations Research*. Springer, 1999.

[68] NVIDIA. CUDA C Programming Guide :: CUDA Toolkit Documentation, November 2012.

[69] Travis E. Oliphant. *Guide to NumPy*. Provo, UT, March 2006.

[70] Tore Opsahl, Filip Agneessens, and John Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32(3):245–251, July 2010.

[71] Johan Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *Photogrammetric Record*, 15(88):589–599, 1996.

[72] Johan Philip. Critical point configurations of the 5-, 6-, 7-, and 8-point algorithms for relative orientation. Technical report, KTH Royal Institute of Technology, 1998.

[73] Clemens Rabe. *Detection of Moving Objects by Spatio-Temporal Motion Analysis: Real-time Motion Estimation for Driver Assistance Systems*. Südwestdeutscher Verlag für Hochschulschriften, 2012. ISBN 978-3-8381-3219-8.

[74] Volker Rodehorst, Matthias Heinrichs, and Olaf Hellwich. Evaluation of relative pose estimation methods for multi-camera setups. In *Proceedings of International Society for Photogrammetry and Remote Sensing*, 2008.

[75] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: a machine learning approach to corner detection. *TPAMI*, 32(1):105–119, 2010.

[76] Konrad Schindler, Andreas Ess, Bastian Leibe, and Luc Van Gool. Automatic detection and tracking of pedestrians from a moving stereo rig. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):523 – 537, 2010.

[77] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR*, volume 94, pages 593–600. IEEE, 1994.

[78] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *CVPR*. IEEE, 2008.

[79] Chanop Silpa-anan and Richard Hartley. Optimised KD-trees for fast image descriptor matching. In *CVPR*, pages 1–8. IEEE, 2008.

[80] Josef Sivic and Andrew Zisserman. Video Google: a text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477. IEEE, 2003.

[81] Noah Snavely, Steven Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[82] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR*, pages 1–8. IEEE, June 2008.

[83] Henrik Stewenius, Christopher Engels, and David Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284 – 294, 2006.

[84] Hauke Strasdat, José M. M. Montiel, and Andrew J. Davison. Visual slam: Why filter? *Image Vision Computing*, 30(2):65–77, February 2012.

[85] Christoph Strecha, Wolfgang von Hansen, Luc Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, pages 1–8. IEEE, June 2008.

[86] Peter Sturm. A historical survey of geometric computer vision. In *CAIP*, pages 1–8. Springer, 2011.

[87] Raghav Subbarao, Yakup Genc, and Peter Meer. Nonlinear Mean Shift for Robust Pose Estimation. In *Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision*, pages 6–6. IEEE, February 2007.

[88] SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2012.

[89] Carlo Tomasi. Detection and Tracking of Point Features Technical Report CMU-CS-91-132. *Image (Rochester, N.Y.)*, (April), 1991.

[90] Philip Torr and Andrew Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15(8):591–605, 1997.

[91] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV, pages 298–372. Springer, 2000.

[92] Christian Unger. Dense 3d reconstruction from multiple views using non-linear least squares optimizations. Masterthesis, Technische Universität Berlin, 2013.

[93] Raquel Urtasun, David Fleet, and Pascal Fua. 3D People Tracking with Gaussian Process Dynamical Models. In *CVPR*, volume 1, pages 238–245. Citeseer, IEEE, 2006.

[94] Cornelius Wefelscheid, Ronny Hänsch, and Olaf Hellwich. Three-dimensional building reconstruction using images obtained by unmanned aerial vehicles. In *Proceedings of the International Conference on Unmanned Aerial Vehicle in Geomatics (UAV-g)*, Zurich, Suisse, Sep 2011.

[95] Cornelius Wefelscheid and Olaf Hellwich. OpenOF: Framework for sparse non-linear least squares optimization on a GPU. In *VISAPP*. SciTePress, 2013.

[96] Cornelius Wefelscheid, Tilman Wekel, and Olaf Hellwich. Monocular rectangle reconstruction - based on direct linear transformation. In *VISAPP*, pages 271–276. SciTePress, 2011.

[97] Changchang Wu. VisualSFM: A Visual Structure from Motion System, December 2012. http://homes.cs.washington.edu/ ccwu/vsfm/.

[98] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven Seitz. Multicore bundle adjustment. In *CVPR*, pages 3057–3064. IEEE, 2011.

[99] Changchang Wu, Brian Clipp, Xiaowei Li, Jan-Michael Frahm, and Marc Pollefeys. 3d model matching with viewpoint-invariant patches (vip). In *CVPR*. IEEE, 2008.

[100] Christopher Zach. Sources, February 2012. http://www.inf.ethz.ch/personal/chzach/opensource.